HZ BOOKS
华章教育

计 算 机 科 学 丛 书

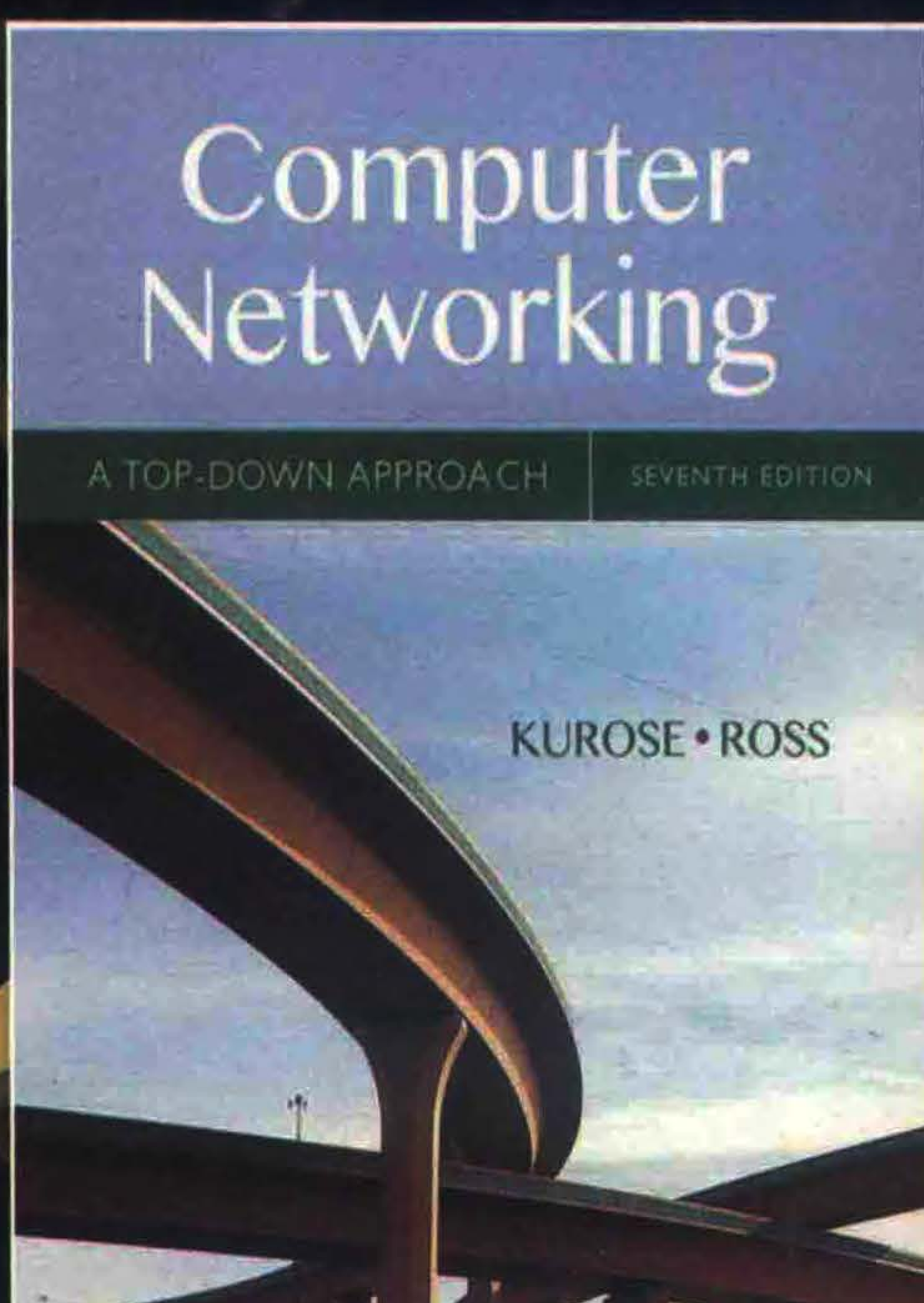
Pearson

原书第7版

计算机网络
自顶向下方法

[美] 詹姆斯 · F. 库罗斯（James F. Kurose） 基思 · W. 罗斯（Keith W. Ross） 著
马萨诸塞大学阿默斯特分校 纽约大学上海分校
陈鸣 译

Computer Networking
A Top-Down Approach Seventh Edition



机械工业出版社
China Machine Press

VISIT…

LANZAROTE Caliente.com

计算机科学丛书

原书第7版

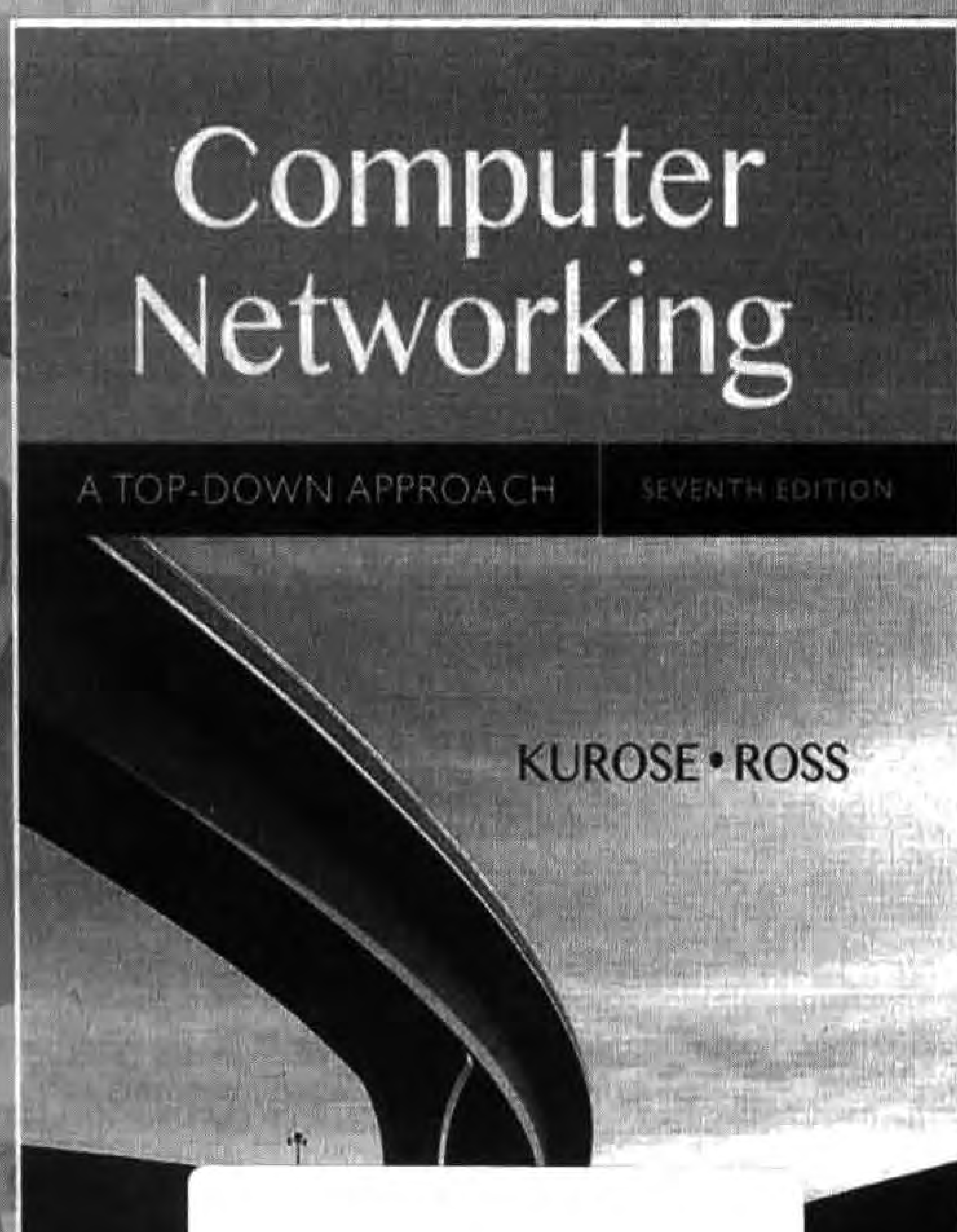
计算机网络

自顶向下方法

[美] 詹姆斯·F. 库罗斯（James F. Kurose） 基思·W. 罗斯（Keith W. Ross） 著
马萨诸塞大学阿默斯特分校 纽约大学上海分校
陈鸣 译

Computer Networking

A Top-Down Approach Seventh Edition



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

计算机网络：自顶向下方法（原书第 7 版）/（美）詹姆斯·F. 库罗斯（James F. Kurose），（美）基思·W. 罗斯（Keith W. Ross）著；陈鸣译．—北京：机械工业出版社，2018.5
（计算机科学丛书）
书名原文：Computer Networking：A Top-Down Approach，Seventh Edition

ISBN 978-7-111-59971-5

I. 计… II. ①詹… ②基… ③陈… III. 计算机网络－教材 IV. TP393

中国版本图书馆 CIP 数据核字（2018）第 104318 号

本书版权登记号：图字 01-2016-4521

Authorized translation from the English language edition, entitled *Computer Networking: A Top-Down Approach, 7E*, 9780133594140 by James F. Kurose and Keith W. Ross, published by Pearson Education, Inc., Copyright © 2017，2013，2010.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Chinese simplified language edition published by China Machine Press Copyright © 2018.

本书中文简体字版由 Pearson Education（培生教育出版集团）授权机械工业出版社在中华人民共和国境内（不包括香港、澳门特别行政区及台湾地区）独家出版发行。未经出版者书面许可，不得以任何方式抄袭、复制或节录本书中的任何部分。

本书封底贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

本书是经典的计算机网络教材，采用作者独创的自顶向下方法来讲授计算机网络的原理及其协议，自第 1 版出版以来已经被数百所大学和学院选作教材，被译为 14 种语言。

第 7 版保持了以前版本的特色，继续关注因特网和计算机网络的现代处理方式，注重原理和实践，为计算机网络教学提供一种新颖和与时俱进的方法。同时，第 7 版进行了相当多的修订和更新，首次改变了各章的组织结构，将网络层分成两章（第 4 章关注网络层的数据平面，第 5 章关注网络层的控制平面），并将网络管理主题放入新的第 5 章中。此外，为了反映自第 6 版以来计算机网络领域的新变化，对其他章节也进行了更新，删除了 FTP 和分布式散列表的材料，用流行的因特网显式拥塞通告（ECN）材料代替了 ATM 网络的材料，更新了有关 802.11（即 WiFi）网络和蜂窝网络（包括 4G 和 LTE）的材料，全面修订并增加了新的课后习题，等等。

本书适合作为计算机、电气工程等专业本科生的“计算机网络”课程教科书，同时也适合网络技术人员、专业研究人员阅读。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：刘立卿 迟振春	责任校对：殷 虹
印 刷：北京市荣盛彩色印刷有限公司	版 次：2018 年 6 月第 1 版第 1 次印刷
开 本：185mm×260mm 1/16	印 张：33
书 号：ISBN 978-7-111-59971-5	定 价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

版权所有·侵权必究
封底无防伪标均为盗版
本书法律顾问：北京大成律师事务所 韩光/邹晓东

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www. hzbook. com
电子邮件：hzjsj@ hzbook. com
联系电话：(010)88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心



James F. Kurose 是美国马萨诸塞大学阿默斯特分校杰出的计算机科学教授。他目前担任美国国家科学基金会的副主任，领导计算机和信息科学工程理事会。

Kurose 博士在教育领域的活动获得了许多赞誉，其中包括国立技术大学(8 次)、马萨诸塞大学和研究生院东北联合会杰出教师奖。他获得了 IEEE Taylor Booth 教育奖章，从而确立了他在马萨诸塞共同体信息技术促进会的领导地位。他多次赢得优秀会议论文奖并获得 IEEE Infocom 成就奖和 ACM Sigcomm 的时间考验奖。

Kurose 博士是《IEEE 通信会刊》(IEEE Transactions on Communications)和《IEEE/ACM 网络会刊》(IEEE/ACM Transactions on Networking)的前任总编辑。他担任了 IEEE Infocom、ACM SIGCOMM、ACM 因特网测量会议和 ACM SIGMETRICS 的技术程序的共同主席。他是 IEEE 会士(Fellow)和 ACM 会士。他的研究兴趣包括网络协议和体系结构、网络测量、多媒体通信以及建模和性能评价。他拥有哥伦比亚大学计算机科学的博士学位。



Keith W. Ross 是美国纽约大学(NYU)上海分校工程和计算机科学学院院长以及 NYU 计算机科学和工程系的 Leonard J. Shustek 首席教授。在此之前，他就职于宾夕法尼亚大学(13 年)、Eurecom 学院(5 年)和理工大学(10 年)。他从 Tufts 大学获得电气工程学士学位，从哥伦比亚大学获得电气工程硕士学位，从密歇根大学获得计算机和控制工程博士学位。Ross 也是 Wimba 公司奠基人和首任 CEO，该公司为电子学习研发了在线多媒体应用并于 2010 年被 Blackboard 收购。

Ross 教授的研究兴趣在隐私、社交网络、对等(P2P)网络、因特网测量、内容分发网和随机建模等方面。他是 ACM 会士和 IEEE 会士，获得了 Infocom 2009 年优秀论文奖，并且获得《多媒体通信》2011 年和 2008 年优秀论文奖(由 IEEE 通信学会授予)。他担任多个杂志编委和会议程序委员会委员，包括《IEEE/ACM 网络会刊》、ACM SIGCOMM、ACM CoNext 和 ACM 因特网测量会议。他还担任联邦贸易委员会 P2P 文件共享方面的顾问。



陈鸣，南京航空航天大学特聘教授、研究生导师；分别于 1982 年、1988 年在解放军信息工程学院获得学士、硕士学位，于 1991 年在解放军通信工程学院获得博士学位，1999 ~ 2000 年为美国哥伦比亚大学访问科学家，现在任中国计算机学会网络与数据通信专委会副主任，是中国通信学会等多个学术团体委员和 IEEE 会员；长期从事网络测量、分布式系统、未来网络、网络安全等领域研究和教学工作，近期研究兴趣包括无人机网络、软件定义网络、网络功能虚拟化；承担了国家自然科学基金、国家 863、国家 973 子课题等项目；开发的多个网络管理系统和应用系统在多个领域得到广泛应用；撰写网络著作近 10 本，发表 SCI/EI 论文几十篇，有国家发明专利 10 项；获得国家教学成果二等奖 1 项和省部级科技进步二、三等奖十几项。

译者序

我自 2003 年起翻译 Kurose 和 Ross 两位教授合著的《计算机网络：自顶向下方法》教科书，至今已经过去 15 个年头了。连续六次翻译这本计算机网络经典教科书为我提供了难得的学习机会，使我能够静下心来思考他们的所思所想，洞察他们对技术发展的分析，品味他们讲解网络原理时的遣词造句。尽管我们能够从多种来源学习、研究网络技术，但是通过这本书，我们能够系统地理解错综复杂的网络技术是如何组织成为复杂的网络系统的，观察到新型网络技术是如何替代过时技术的，感受到一些网络术语是如何悄然变化的，学习大师级人物是如何创新网络教学的……总之，这本教科书被称为世界上非常流行、非常优秀的网络教科书的确当之无愧。同时，该书的中译本有助于我国高校学生、科技工作者以及其他读者高效地学习计算机网络知识(而不是英文!)，进一步促进我国网络教学水平的提升。

软件定义网络(SDN)的发展是网络界近年来非常重要和令人兴奋的事件，本书首次用两章的篇幅讨论网络层——第 4 章讨论网络层的数据平面，第 5 章讨论网络层的控制平面和网络管理主题。与以前的版本一样，本书与时俱进地更新了许多章节的内容、习题和网站的配套资料，反映了网络技术日新月异的发展。

在本书的翻译过程中，译者得到了南京航空航天大学计算机科学与技术学院的领导和同事的支持和帮助。本书的责任编辑刘立卿、迟振春出色的专业技能和耐心细致也使本书增色。

限于时间和学识，译文错漏难免，请识者不吝赐教。请将问题发送给 mingchennj@163.com，我将及时反馈读者。

陈鸣
南京航空航天大学计算机科学与技术学院

欢迎阅读《计算机网络：自顶向下方法》的第 7 版。自从本书第 1 版于 16 年前出版以来，这本书已经被数百所大学和学院采用，被译为 14 种语言，并被世界上几十万的学生和从业人员使用。我们倾听了许多读者的意见，赞扬之声不绝于耳。

第 7 版的新颖之处

我们认为本书成功的一个重要原因是，持续地为计算机网络教学提供了一种新颖和与时俱进的方法。在第 7 版中，我们做了不少改变，但也保持了我们认为(并且得到了使用本书的教师和学生的认可)本书最为重要的方面：它的自顶向下方法，它关注因特网和计算机网络的现代处理方式，它的注重原理和实践，以及它易于理解的风格和学习计算机网络的方法。然而，第 7 版进行了相当多的修订和更新。

本书的长期读者将注意到，自该教科书出版以来，我们首次改变了各章的组织结构。以往包括在一章中的网络层，现在包括在第 4 章(该章关注网络层的所谓“数据平面”组件)和第 5 章(该章关注网络层的“控制平面”)中。网络层范围的扩展反映了软件定义网络(SDN)重要性的迅速提升，无可置疑地证明了十年来网络非常重要和令人兴奋的进展。尽管 SDN 是相对新近的创新，但它已经迅速在实践中得到应用，所以介绍现代计算机网络而不涉及 SDN 已经成为不可想象的事。先前包括在第 9 章中的网络管理主题，现在已经放入新的第 5 章中了。我们一如既往地更新了本书的许多其他章节，以反映自第 6 版以来网络领域的新变化。从教科书中撤除的材料一如既往地能够在本书的配套网站中找到。比较重要的一些更新如下：

- 为反映因特网不断增长的范围和应用，更新了第 1 章。
- 讨论应用层的第 2 章进行了重大更新。我们删除了 FTP 和分布式散列表的材料，增加了有关应用层视频流和内容分发网，以及 Netflix 和 YouTube 学习案例的新节。套接字编程已从 Python 2 更新为 Python 3。
- 讨论运输层的第 3 章进行了适度更新。异步传递方式(ATM)网络的材料已被更流行的因特网显式拥塞通告(ECN)的材料所代替，用 ECN 教授相同的原理。
- 第 4 章讨论网络层的“数据平面”组件。数据平面是每台路由器的转发功能，它决定到达路由器的一条输入链路的分组是如何转发到该路由器的一条输出链路的。我们更新了在所有前面版本中都能找到的有关传统因特网转发的材料，并且增加了分组调度的材料。还增加了有关通用转发的新节，就像 SDN 中实践的那样。该章也有很多更新。有关多播和广播通信方面的材料已被删除。
- 第 5 章讨论网络层的控制平面功能。这种网络范围的逻辑控制着数据报如何沿着从源主机到目的主机的路由器的端到端路径选择路由。如同前面版本一样，我们包括路由选择算法以及在今天的因特网中所使用的路由选择协议(更新了对 BGP 的讨论)。增加了有关 SDN 控制平面的新节，讲述了如何在所谓的 SDN 控制器中实现路由选择和其他功能。

- 第 7 章讨论无线网络和移动网络，更新了有关 802．11(所谓 WiFi)网络和蜂窝网络(包括 4G 和 LTE)的材料。
- 第 8 章讨论网络安全，这在第 6 版中已经全面更新过了，本版中该章仅进行了适度更新。
- 第 9 章较之第 6 版进行了适度调整，有关视频流和内容分发网的材料已经放到第 2 章了，有关分组调度的材料已经放进第 4 章了。
- 增加了与每章后面习题相关的重要新材料。与前面各版一样，对课后习题进行了修订、增加和删除。

在新版本的修订过程中，我们的目标一如既往，继续关注因特网和计算机网络的现代处理方式，注重原理和实践。

本书读者对象

本书适用于计算机网络的第一门课程，既可用于计算机科学系的学生，也可用于电气工程系的学生。就编程语言而言，本书仅假定学生具有 C、C++、Java 或 Python 的编程经验(也只是在几个地方用到)。与许多入门性的其他计算机网络教科书相比，尽管本书表述更为精确，分析更为细致，然而书中很少用到高中阶段没有教过的数学概念。我们有意避免使用任何高等微积分、概率论或随机过程的概念(尽管我们为具有这种高级背景的学生准备了某些课后习题)。因此，本书适用于本科生课程和一年级研究生课程。它对于电信业的从业人员也应当有用。

本书的独特之处

计算机网络这门课程极为复杂，涉及许多以错综复杂的方式彼此交织的概念、协议和技术。为了处理这种大跨度和高复杂性，许多计算机网络教科书都围绕计算机网络体系结构的“层次”来组织内容。借助于这种分层的组织结构，学生能够透过计算机网络的复杂性看到其内部，他们在学习整个体系结构的某个部分中的独特概念和协议的同时，也能看清所有这些部分如何整合在一起的全貌。从教学法的角度来看，我们的个人体验是这种分层的教学方法的确是卓有成效的。尽管如此，我们发现那种自底向上的传统教学方法，即从物理层到应用层逐层进行讲解的方法，对于现代计算机网络课程并非是最佳的方法。

自顶向下方法

本书于 16 年前首次以自顶向下的方式来对待网络，这就是说从应用层开始向下一直讲到物理层。我们从教师以及学生那里得到的反馈证实了这种自顶向下方法有许多好处，并且从教学法来讲的确很好实施。第一，它特别强调应用层(它是网络中的“高增长领域”)。的确，计算机网络中的许多近期革命都发生在应用层，其中包括 Web、对等文件共享和媒体流。及早强调应用层的问题与大多数其他教科书中所采取的方法不同，那些教科书中只有少量有关网络应用、网络应用的需求、应用层范式(例如客户-服务器和对等方到对等方)以及应用编程接口方面的内容。第二，我们(和使用本书的许多教师)作为教师的经验是，在课程开始后就教授网络应用的内容，是一种有效激励学习积极性的工具。学生急于知道诸如电子邮件和 Web 等网络应用是如何工作的，这些应用是多数学生每天都

在使用的东西。一旦理解了这些应用，学生便能够理解支持这些应用的网络服务，接下来则会仔细思考在较低层次中可能提供和实现这些服务的各种方式。因此，及早地涉及应用程序能够激发学生学习本书其余部分的积极性。

第三，自顶向下方法使得教师能够在教学的早期阶段介绍网络应用程序的开发。学生不仅能够明白流行的应用程序和协议的工作原理，还能学习到创造自己的网络应用程序和应用级协议是多么容易。采用自顶向下的方法后，学生能够及早地搞清楚套接字编程、服务模型和协议的概念，这些重要概念为后续各层的讨论做了铺垫。通过提供用 Python 语言写成的套接字编程的例子，我们强调主要思想，而不会使学生陷于复杂代码的困境。电气工程和计算机科学系的本科生理解这些代码应当不会有困难。

以因特网为研究目标

尽管自第 4 版起我们从书名中去掉了"Featuring the Internet"（描述因特网特色）这个短语，但这并不意味着我们不再关注因特网！的确，一切如初！而且由于因特网已经变得无所不在，我们反而认为任何网络教科书都必须非常关注因特网，因此该短语在某种程度上已经没有必要了。我们继续使用因特网的体系结构和协议作为基本载体来学习基本的计算机网络概念。当然，我们也能把概念和协议放入其他网络体系结构中讲解。但是我们的关注焦点是因特网，这反映在我们围绕因特网体系结构的 5 层模型来组织材料，这 5 个层次是应用层、运输层、网络层、链路层和物理层。

聚焦因特网的另一个好处是，大多数计算机科学和电气工程的学生急切地希望学习因特网及其协议。他们知道因特网是一种革命性和破坏性的技术，正在深刻地改变着我们的世界。有了对因特网大量中肯的认识后，学生自然而然会对学习其内部原理有了求知欲。因此，教师用因特网作为导向焦点，就易于调动学生学习基本原理的积极性了。

教授网络原理

本书的两个独特之处是自顶向下方法和关注因特网，这已经显现在本书的书名中。如果我们在副书名中强行增加第三个词的话，*原理*一词将包括在其中。网络领域已经发展得相当成熟，能够认识清楚许多基础性的重要问题。例如，在运输层，基础性问题包括建立在不可靠的网络层上的可靠通信、连接建立/拆除与握手、拥塞和流量控制以及多路复用。三个非常重要的网络层问题是，在两台路由器之间找到"好的"路径、互联大量的异构网络和管理现代网络的复杂性。在链路层，基础性问题是共享多路访问信道。在网络安全中，提供机密性、鉴别和报文完整性的技术都基于密码学基本理论。本书在指明基础性网络问题的同时，也会介绍解决这些问题的方法。学习这些原理的学生将获得具有长"保质期"的知识，在今天的网络标准和协议已经变得过时后的很长时间，其中的原理将仍然重要和中肯。我们相信，用因特网将学生引入网络之门后，再强调基础性问题及其解决方案，这种两者结合的方法将使他们迅速理解几乎任何网络技术。

Web 站点

本书配套的 Web 站点位于 http://www.pearsonhighered.com/cs-resources/。该站点包括：

- 交互式学习材料。本书的配套网站包括视频要点(VideoNotes)，即由作者制作的全书重要主题的视频呈现，以及对习题解答的简要讲解，这些习题类似于每章后面的习题。我们已经在 Web 站点上提供了第1~5 章的视频要点和在线习题，并且将随着时间的推移而继续积极地增加和更新这些材料。如在前面版本中一样，该 Web 站点包含了交互式 Java 小程序，以动画方式显示了重要的网络概念。该站点也包括交互式小测验，允许学生检查他们对该专题内容的基本理解。教授能够将这些交互式特色结合到他们的讲义中或将它们用作小实验。
- 附加的技术材料。由于我们在本书的每个版本中都增加了新材料，故我们不得不删去某些现有主题以保持篇幅的合理。例如，为了在本版中为新材料腾出空间，我们删除了有关 FTP、分布式散列表和多播方面的材料。出现在本书较早版本中的材料仍然是有益的，并且能够在本书 Web 网站上找到。
- 编程作业。Web 网站也提供了一些详细的编程作业，这些编程作业包括构建一台多线程 Web 服务器，构建一个具有图形用户接口(GUI)的电子邮件客户，以及发送端和接收端可靠数据传输协议的编程，分布式路由选择算法的编程，等等。
- Wireshark 实验。通过观察网络协议的实际运行，读者能够大大加深对它们的理解。该 Web 站点提供了许多 Wireshark 作业，使学生能够实际观察两个协议实体之间报文的交换顺序。该 Web 站点包括了有关 HTTP、DNS、TCP、UDP、IP、ICMP、以太网、ARP、WiFi 和 SSL 的单独 Wireshark 实验，以及在跟踪一个获取 Web 网页的请求时所涉及的所有协议的 Wireshark 实验。随着时间的推移，我们将继续增加新的实验。

除了配套网站外，作者维护了一个公共网站 http://gaia.cs.umass.edu/kurose_ross/interactive，该网站包括交互式练习，这些练习能够生成与每章后面所选习题类似的问题(并给出解答)。由于学生能够产生无数类似问题的实例并看到解答，所以他们能够练习到真正掌握为止。

教学特色

我们每位作者都教了 30 多年的计算机网络课程，这本书凝聚了我们总共 60 多年教了几千名学生的教学经验。在此期间，我们也成为计算机网络领域活跃的研究人员。(事实上，James 和 Keith 于 1979 年在哥伦比亚大学彼此相识，共同选了由 Mischa Schwartz 执教的硕士研究生计算机网络课程。) 所有这些都让我们对网络现状和网络未来的可能发展方向有了良好的观察力。无论如何，我们在组织这本书的材料时，抵御住了偏向自己所钟爱的研究项目的诱惑。如果你对我们的研究工作感兴趣的话，可以访问我们的个人 Web 网站。因此，这是一本关于现代计算机网络的书，即该书包含了当代协议和技术以及支撑这些协议和技术的基本原理。我们也认为学习（和讲授）网络是令人开心的事。本书中包括的幽默、使用的类比和现实世界的例子将有望使这些材料更具趣味性。

教师的补充材料[㊀]

我们提供了一套完整的补充材料，以帮助教师教授这门课程。这些材料都能通过访问 Pearson 的教师资源中心（http://www. pearsonhighered. com/irc）得到。有关获取这些教师补充材料的信息可访问教师资源中心。

- *PowerPoint 幻灯片*。我们提供了全部 9 章的 PowerPoint 幻灯片。这些幻灯片根据第 7 版进行了彻底更新，详细地涵盖了每章的内容。幻灯片中使用了图片和动画（而非仅是单调的文本标题），这使得它们有趣且在视觉上有吸引力。我们向教师提供了原始的幻灯片，使得教师能够做个性化修改以满足自己的教学需要。这些幻灯片中的某些部分就是由采用本书进行教学的教师所贡献的。
- *课后习题解答*。我们提供了本书中课后习题的解题手册、编程作业和 Wireshark 实验。如前所述，我们在本书的前 6 章中引入了许多新的课后作业。

各章间的关联性

本书的第 1 章提供了对计算机网络自包含的概述。该章介绍了许多重要的概念与术语，为本书的其余部分奠定了基础。其他所有章都直接依赖于第 1 章的内容。在讲解完第 1 章之后，我们推荐按顺序讲解第 2 ~6 章的内容，这样就遵循了自顶向下的原则。第 2 ~ 6 章中每一章都会用到前面章节的内容。在完成前 6 章的教学后，教师就有了相当大的灵活性。最后 3 章之间没有任何相关性，因此能够以任何顺序进行教学。然而，最后 3 章中的每一章都依赖于前 6 章中的材料。许多教师采用的教学方案是：教授前 6 章，然后讲授后 3 章之一作为点睛之笔。

最后的话：我们乐于听取你的意见

我们鼓励学生和教师向我们发送电子邮件，发表对本书的任何评论。对我们而言，能够听到来自全世界的教师和学生就本书前 6 版的反馈，是件令人愉快的事。我们已经在本书新版中综合进许多条建议。我们也鼓励教师向我们发送新的课后习题（及其解答），这将完善当前的课后习题。我们将这些习题放在配套 Web 网站上只有教师才能访问的区域。我们也鼓励教师和学生编写新的 Java 小程序来诠释书中的概念和协议。如果你有了认为适合于本书的小程序，请将它发送给作者。如果该小程序（包括标记和术语）合适的话，我们很乐意将它放在本书的网站上，并附上对该小程序作者的适当推荐。

正如谚语所说："让那些卡片和信件到来吧！"我们郑重宣布，请大家一如既往地告诉我们有趣的 URL，指出排版错误，说出不赞成我们的哪些主张，告诉我们怎样做效果好、怎样做效果不好，以及你认为在本书下一版中应当包括哪些内容、删除哪些内容。我们的电子邮件地址是 kurose@ cs. umass. edu 和 ross@ poly. edu。

㊀ 关于本书教辅资源，只有使用本书作为教材的教师才可以申请，需要的教师请联系机械工业出版社华章公司，电话 136 0115 6823，邮箱 wangguang@ hzbook. com。——编辑注

致谢

从 1996 年我们开始撰写本书以来，许多人为我们提供了非常宝贵的帮助，在如何最好地组织和讲授网络课程方面对我们的构思产生了很大影响。在此，我们要对那些从本书的最早书稿到本次第 7 版帮助过我们的所有人道谢，非常感谢大家。我们还要感谢来自世界各地成千上万的读者，包括学生、教职员和从业人员，他们给了我们对于本书以前版本的看法和评论以及对未来版本的建议。特别感谢下列这些人：

Al Aho（哥伦比亚大学）
Hisham Al-Mubaid（休斯敦净湖大学）
Pratima Akkunoor（亚利桑那州立大学）
Paul Amer（特拉华大学）
Shamiul Azom（亚利桑那州立大学）
Lichun Bao（加利福尼亚大学欧文分校）
Paul Barford（威斯康星大学）
Bobby Bhattacharjee（马里兰大学）
Steven Bellovin（哥伦比亚大学）
Pravin Bhagwat（Wibhu）
Supratik Bhattacharyya（前 Sprint 公司人员）
Ernst Biersack（Eurécom 学院）
Shahid Bokhari（工程技术大学 Lahore 分校）
Jean Bolot（Technicolor Research）
Daniel Brushteyn（前宾夕法尼亚大学学生）
Ken Calvert（肯塔基大学）
Evandro Cantu（Santa Catarina 联邦大学）
Jeff Case（SNMP 国际研究院）
Jeff Chaltas（Sprint 公司）
Vinton Cerf（Google 公司）
Byung Kyu Choi（密歇根技术大学）
Bram Cohen（BitTorrent 公司）
Constantine Coutras（培斯大学）
John Daigle（密西西比大学）
Edmundo A. de Souza e Silva（Rio de Janiero 联邦大学）
Philippe Decuetos（Eurécom 学院）
Christophe Diot（Technicolor Research）
Prithula Dhunghel（Akamai 公司）
Deborah Estrin（加利福尼亚大学洛杉矶分校）
Michalis Faloutsos（加利福尼亚大学河滨分校）
Wu-chi Feng（俄勒冈研究生院）
Sally Floyd（ICIR，加利福尼亚大学伯克利分校）
Paul Francis（Max Planck 学院）
Lixin Gao（马萨诸塞大学）
JJ Garcia-Luna-Aceves（加利福尼亚大学圣克鲁兹分校）
Mario Gerla（加利福尼亚大学洛杉矶分校）
David Goodman（纽约大学理工学院）
Yang Guo（Alcatel/Lucent 贝尔实验室）
Tim Griffin（剑桥大学）
Max Hailperin（Gustavus Adolphus 学院）
Bruce Harvey（佛罗里达 A&M 大学，佛罗里达州立大学）
Carl Hauser（华盛顿州立大学）
Rachelle Heller（乔治·华盛顿大学）
Phillipp Hoschka（INRIA/W3C）
Wen Hsin（公园大学）
Albert Huang（前宾夕法尼亚大学学生）
Cheng Huang（微软研究院）
Esther A. Hughes（弗吉尼亚联邦大学）
Van Jacobson（施乐公司帕洛阿尔托研究中心）
Pinak Jain（前纽约大学理工学院学生）
Jobin James（加利福尼亚大学河滨分校）
Sugih Jamin（密歇根大学）
Shivkumar Kalyanaraman（IBM 印度研究院）
Jussi Kangasharju（赫尔辛基大学）
Sneha Kasera（犹他大学）
Parviz Kermani（原在 IBM 研究院）
Hyojin Kim（前宾夕法尼亚大学学生）
Leonard Kleinrock（加利福尼亚大学洛杉矶

分校）
David Kotz（达特茅斯学院）
Beshan Kulapala（亚利桑那州立大学）
Rakesh Kumar（Bloomberg）
Miguel A. Labrador（南佛罗里达大学）
Simon Lam（得克萨斯大学）
Steve Lai（俄亥俄州立大学）
Tom LaPorta（宾夕法尼亚州立大学）
Tim-Berners Lee（万维网研究所）
Arnaud Legout（INRIA）
Lee Leitner（Drexel 大学）
Brian Levine（马萨诸塞大学）
Chunchun Li（前纽约大学理工学院学生）
Yong Liu（纽约大学理工学院）
William Liang（前宾夕法尼亚大学学生）
Willis Marti（得克萨斯 A&M 大学）
Nick McKeown（斯坦福大学）
Josh McKinzie（公园大学）
Deep Medhi（密苏里大学堪萨斯市分校）
Bob Metcalfe（国际数据集团）
Sue Moon（KAIST）
Jenni Moyer（Comcast 公司）
Erich Nahum（IBM 研究院）
Christos Papadopoulos（科罗拉多州立大学）
Craig Partridge（BBN 技术）
Radia Perlman（Sun 公司）
Jitendra Padhye（微软研究院）
Vern Paxson（加利福尼亚大学伯克利分校）
Kevin Phillips（Sprint 公司）
George Polyzos（雅典经济和商业大学）
Sriram Rajagopalan（亚利桑那州立大学）
Ramachadran Ramjee（微软研究院）
Ken Reek（罗切斯特技术学院）
Martin Reisslein（亚利桑那州立大学）
Jennifer Rexford（普林斯顿大学）
Leon Reznik（罗切斯特技术学院）
Pablo Rodrigez（Telefonica）
Sumit Roy（华盛顿大学）
Avi Rubin（约翰斯霍普金斯大学）
Dan Rubenstein（哥伦比亚大学）
Douglas Sanane（John Jay 学院）
Despina Saparilla（思科系统公司）
John Schanz（Comcast 公司）
Henning Schulzrinne（哥伦比亚大学）
Mischa Schwartz（哥伦比亚大学）
Ardash Sethi（特拉华大学）
Harish Sethu（德雷塞尔大学）
K. Sam Shanmugan（堪萨斯大学）
Prashant Shenoy（马萨诸塞大学）
Clay Shields（乔治顿大学）
Subin Shrestra（宾夕法尼亚大学）
Bojie Shu（前纽约大学理工学院）
Mihail L. Sichitiu（北卡罗来纳州立大学）
Peter Steenkiste（卡内基－梅隆大学）
Tatsuya Suda（加利福尼亚大学欧文分校）
Kin Sun Tam（纽约州立大学奥尔巴尼分校）
Don Towsley（马萨诸塞大学）
David Turner（加州州立大学圣贝纳迪诺分校）
Nitin Vaidya（伊利诺斯大学）
Michele Weigle（克莱姆森大学）
David Wetherall（华盛顿大学）
Ira Winston（宾夕法尼亚大学）
吴迪（中山大学）
Shirley Wynn（纽约大学理工学院）
Raj Yavatkar（Intel 公司）
Yechiam Yemini（哥伦比亚大学）
Ming Yu（纽约州立大学宾汉姆顿分校）
Ellen Zegura（佐治亚理工学院）
Honggang Zhang（Suffolk 大学）
Hui Zhang（卡内基－梅隆大学）
Lixia Zhang（加利福尼亚大学洛杉矶分校）
Meng Zhang（前纽约大学理工学院学生）
Shuchun Zhang（前宾夕法尼亚大学学生）
Xiaodong Zhang（俄亥俄州立大学）
ZhiLi Zhang（明尼苏达大学）
Phil Zimmermann（独立顾问）
Mike Zink（马萨诸塞大学）
Cliff C. Zou（中央佛罗里达大学）

我们也要感谢整个培生团队，特别感谢 Matt Goldstein 和 Joanne Manning，他们对本书第 7 版做出了十分杰出的工作（并且他们容忍了两位非常挑剔的作者，在作者看来是根本不可能的最后期限内完成了工作）。感谢两位艺术家 Janet Theurer 和 Patrice Rossi Calkin 为本版和之前版本的优美插图所做的工作，还要感谢 Katie Ostler 以及她在 Cenveo 的团队对本版图书的出色生产工作。最后，特别感谢本书的前两任编辑——Addison-Wesley 出版公司的 Michael Hirsch 和 Susan Hartman。没有他们的有效管理、不断鼓励，以及近乎无限的耐心、幽默和坚定不移，本书将不会达到现在的水平（并且根本不会如此出色）。

目 录

Computer Networking: A Top-Down Approach, Seventh Edition

出版者的话

作译者简介

译者序

前言

第1章 计算机网络和因特网 …… 1

1.1 什么是因特网 …… 1

1.1.1 具体构成描述 …… 1

1.1.2 服务描述 …… 4

1.1.3 什么是协议 …… 5

1.2 网络边缘 …… 6

1.2.1 接入网 …… 8

1.2.2 物理媒体 …… 13

1.3 网络核心 …… 15

1.3.1 分组交换 …… 15

1.3.2 电路交换 …… 19

1.3.3 网络的网络 …… 22

1.4 分组交换网中的时延、丢包和吞吐量 …… 24

1.4.1 分组交换网中的时延概述 …… 24

1.4.2 排队时延和丢包 …… 27

1.4.3 端到端时延 …… 28

1.4.4 计算机网络中的吞吐量 …… 30

1.5 协议层次及其服务模型 …… 32

1.5.1 分层的体系结构 …… 32

1.5.2 封装 …… 36

1.6 面对攻击的网络 …… 37

1.7 计算机网络和因特网的历史 …… 40

1.7.1 分组交换的发展：1961～1972 …… 41

1.7.2 专用网络和网络互联：1972～1980 …… 42

1.7.3 网络的激增：1980～1990 …… 42

1.7.4 因特网爆炸：20世纪90年代 …… 43

1.7.5 最新发展 …… 44

1.8 小结 …… 44

课后习题和问题 …… 46

复习题 …… 46

习题 …… 47

Wireshark 实验 …… 51

人物专访 …… 52

第2章 应用层 …… 54

2.1 应用层协议原理 …… 54

2.1.1 网络应用程序体系结构 …… 55

2.1.2 进程通信 …… 57

2.1.3 可供应用程序使用的运输服务 …… 59

2.1.4 因特网提供的运输服务 …… 60

2.1.5 应用层协议 …… 63

2.1.6 本书涉及的网络应用 …… 63

2.2 Web 和 HTTP …… 64

2.2.1 HTTP 概况 …… 64

2.2.2 非持续连接和持续连接 …… 65

2.2.3 HTTP 报文格式 …… 67

2.2.4 用户与服务器的交互：cookie …… 70

2.2.5 Web 缓存 …… 72

2.2.6 条件 GET 方法 …… 74

2.3 因特网中的电子邮件 …… 75

2.3.1 SMTP …… 76

2.3.2 与 HTTP 的对比 …… 78

2.3.3 邮件报文格式 …… 79

2.3.4 邮件访问协议 …… 79

2.4 DNS：因特网的目录服务 …… 83

2.4.1 DNS 提供的服务 …… 83

2.4.2 DNS 工作机理概述 …… 85

2.4.3 DNS 记录和报文 …… 89

2.5 P2P 文件分发 …… 92

2.6 视频流和内容分发网 …… 97

2.6.1 因特网视频 …… 97

2.6.2 HTTP 流和 DASH …… 98

2.6.3 内容分发网 …… 98

2.6.4 学习案例：Netflix、YouTube 和“看看” …… 101

2.7 套接字编程：生成网络应用 …… 104

2.7.1 UDP 套接字编程 …… 105
2.7.2 TCP 套接字编程 …… 109
2.8 小结 …… 112
课后习题和问题 …… 113
复习题 …… 113
习题 …… 114
套接字编程作业 …… 118
Wireshark 实验：HTTP …… 119
Wireshark 实验：DNS …… 120
人物专访 …… 120
第 3 章 运输层 …… 121
3.1 概述和运输层服务 …… 121
3.1.1 运输层和网络层的关系 …… 122
3.1.2 因特网运输层概述 …… 123
3.2 多路复用与多路分解 …… 125
3.3 无连接运输：UDP …… 130
3.3.1 UDP 报文段结构 …… 132
3.3.2 UDP 检验和 …… 133
3.4 可靠数据传输原理 …… 134
3.4.1 构造可靠数据传输协议 …… 135
3.4.2 流水线可靠数据传输协议 …… 143
3.4.3 回退 N 步 …… 145
3.4.4 选择重传 …… 148
3.5 面向连接的运输：TCP …… 152
3.5.1 TCP 连接 …… 152
3.5.2 TCP 报文段结构 …… 154
3.5.3 往返时间的估计与超时 …… 157
3.5.4 可靠数据传输 …… 159
3.5.5 流量控制 …… 164
3.5.6 TCP 连接管理 …… 166
3.6 拥塞控制原理 …… 170
3.6.1 拥塞原因与代价 …… 171
3.6.2 拥塞控制方法 …… 175
3.7 TCP 拥塞控制 …… 176
3.7.1 公平性 …… 183
3.7.2 明确拥塞通告：网络辅助拥塞控制 …… 184
3.8 小结 …… 185
课后习题和问题 …… 187
复习题 …… 187
习题 …… 189
编程作业 …… 195
Wireshark 实验：探究 TCP …… 196
Wireshark 实验：探究 UDP …… 196
人物专访 …… 196
第 4 章 网络层：数据平面 …… 198
4.1 网络层概述 …… 198
4.1.1 转发和路由选择：数据平面和控制平面 …… 199
4.1.2 网络服务模型 …… 202
4.2 路由器工作原理 …… 203
4.2.1 输入端口处理和基于目的地转发 …… 205
4.2.2 交换 …… 207
4.2.3 输出端口处理 …… 209
4.2.4 何处出现排队 …… 209
4.2.5 分组调度 …… 211
4.3 网际协议：IPv4、寻址、IPv6 及其他 …… 214
4.3.1 IPv4 数据报格式 …… 214
4.3.2 IPv4 数据报分片 …… 216
4.3.3 IPv4 编址 …… 217
4.3.4 网络地址转换 …… 225
4.3.5 IPv6 …… 227
4.4 通用转发和 SDN …… 231
4.4.1 匹配 …… 233
4.4.2 动作 …… 234
4.4.3 匹配加动作操作中的 OpenFlow 例子 …… 234
4.5 小结 …… 236
课后习题和问题 …… 236
复习题 …… 236
习题 …… 237
Wireshark 实验 …… 240
人物专访 …… 241
第 5 章 网络层：控制平面 …… 242
5.1 概述 …… 242
5.2 路由选择算法 …… 244
5.2.1 链路状态路由选择算法 …… 246
5.2.2 距离向量路由选择算法 …… 248
5.3 因特网中自治系统内部的路由选择：OSPF …… 254

5.4 ISP之间的路由选择：BGP ········· 256
5.4.1 BGP的作用 ········· 257
5.4.2 通告BGP路由信息 ········· 257
5.4.3 确定最好的路由 ········· 259
5.4.4 IP任播 ········· 261
5.4.5 路由选择策略 ········· 262
5.4.6 拼装在一起：在因特网中呈现 ········· 264
5.5 SDN控制平面 ········· 265
5.5.1 SDN控制平面：SDN控制器和SDN网络控制应用程序 ········· 266
5.5.2 OpenFlow协议 ········· 267
5.5.3 数据平面和控制平面交互的例子 ········· 269
5.5.4 SDN的过去与未来 ········· 270
5.6 ICMP：因特网控制报文协议 ········· 272
5.7 网络管理和SNMP ········· 274
5.7.1 网络管理框架 ········· 274
5.7.2 简单网络管理协议 ········· 275
5.8 小结 ········· 277
课后习题和问题 ········· 278
复习题 ········· 278
习题 ········· 279
套接字编程作业 ········· 281
编程作业 ········· 282
Wireshark实验 ········· 282
人物专访 ········· 283
第6章 链路层和局域网 ········· 285
6.1 链路层概述 ········· 285
6.1.1 链路层提供的服务 ········· 287
6.1.2 链路层在何处实现 ········· 287
6.2 差错检测和纠正技术 ········· 288
6.2.1 奇偶校验 ········· 289
6.2.2 检验和方法 ········· 290
6.2.3 循环冗余检测 ········· 291
6.3 多路访问链路和协议 ········· 292
6.3.1 信道划分协议 ········· 294
6.3.2 随机接入协议 ········· 295
6.3.3 轮流协议 ········· 301
6.3.4 DOCSIS：用于电缆因特网接入的链路层协议 ········· 301
6.4 交换局域网 ········· 302
6.4.1 链路层寻址和ARP ········· 303
6.4.2 以太网 ········· 308
6.4.3 链路层交换机 ········· 312
6.4.4 虚拟局域网 ········· 317
6.5 链路虚拟化：网络作为链路层 ········· 319
6.6 数据中心网络 ········· 322
6.7 回顾：Web页面请求的历程 ········· 326
6.7.1 准备：DHCP、UDP、IP和以太网 ········· 326
6.7.2 仍在准备：DNS和ARP ········· 327
6.7.3 仍在准备：域内路由选择到DNS服务器 ········· 328
6.7.4 Web客户-服务器交互：TCP和HTTP ········· 329
6.8 小结 ········· 330
课后习题和问题 ········· 331
复习题 ········· 331
习题 ········· 331
Wireshark实验 ········· 335
人物专访 ········· 336
第7章 无线网络和移动网络 ········· 338
7.1 概述 ········· 339
7.2 无线链路和网络特征 ········· 341
7.3 WiFi：802.11无线LAN ········· 346
7.3.1 802.11体系结构 ········· 347
7.3.2 802.11 MAC协议 ········· 350
7.3.3 IEEE 802.11帧 ········· 353
7.3.4 在相同的IP子网中的移动性 ········· 355
7.3.5 802.11中的高级特色 ········· 356
7.3.6 个人域网络：蓝牙和ZigBee ········· 357
7.4 蜂窝因特网接入 ········· 358
7.4.1 蜂窝网体系结构概述 ········· 359
7.4.2 3G蜂窝数据网：将因特网扩展到蜂窝用户 ········· 360
7.4.3 走向4G：LTE ········· 362
7.5 移动管理：原理 ········· 364
7.5.1 寻址 ········· 367
7.5.2 路由选择到移动节点 ········· 367
7.6 移动IP ········· 371

7.7 管理蜂窝网中的移动性 …… 374
7.7.1 对移动用户呼叫的路由选择 …… 375
7.7.2 GSM 中的切换 …… 376
7.8 无线和移动性：对高层协议的影响 …… 378
7.9 小结 …… 380
课后习题和问题 …… 380
复习题 …… 380
习题 …… 381
Wireshark 实验 …… 383
人物专访 …… 383
第 8 章 计算机网络中的安全 …… 385
8.1 什么是网络安全 …… 385
8.2 密码学的原则 …… 387
8.2.1 对称密钥密码体制 …… 388
8.2.2 公开密钥加密 …… 392
8.3 报文完整性和数字签名 …… 396
8.3.1 密码散列函数 …… 397
8.3.2 报文鉴别码 …… 398
8.3.3 数字签名 …… 399
8.4 端点鉴别 …… 404
8.4.1 鉴别协议 ap1.0 …… 404
8.4.2 鉴别协议 ap2.0 …… 405
8.4.3 鉴别协议 ap3.0 …… 405
8.4.4 鉴别协议 ap3.1 …… 406
8.4.5 鉴别协议 ap4.0 …… 406
8.5 安全电子邮件 …… 407
8.5.1 安全电子邮件 …… 407
8.5.2 PGP …… 409
8.6 使 TCP 连接安全：SSL …… 410
8.6.1 宏观描述 …… 411
8.6.2 更完整的描述 …… 413
8.7 网络层安全性：IPsec 和虚拟专用网 …… 415
8.7.1 IPsec 和虚拟专用网 …… 415
8.7.2 AH 协议和 ESP 协议 …… 416
8.7.3 安全关联 …… 416
8.7.4 IPsec 数据报 …… 417
8.7.5 IKE：IPsec 中的密钥管理 …… 420
8.8 使无线 LAN 安全 …… 420
8.8.1 有线等效保密 …… 421
8.8.2 IEEE 802.11i …… 422
8.9 运行安全性：防火墙和入侵检测系统 …… 424
8.9.1 防火墙 …… 424
8.9.2 入侵检测系统 …… 429
8.10 小结 …… 431
课后习题和问题 …… 432
复习题 …… 432
习题 …… 434
Wireshark 实验 …… 437
IPsec 实验 …… 437
人物专访 …… 438
第 9 章 多媒体网络 …… 439
9.1 多媒体网络应用 …… 439
9.1.1 视频的性质 …… 439
9.1.2 音频的性质 …… 440
9.1.3 多媒体网络应用的类型 …… 441
9.2 流式存储视频 …… 443
9.2.1 UDP 流 …… 444
9.2.2 HTTP 流 …… 444
9.3 IP 语音 …… 447
9.3.1 尽力而为服务的限制 …… 448
9.3.2 在接收方消除音频的时延抖动 …… 449
9.3.3 从丢包中恢复 …… 451
9.3.4 学习案例：使用 Skype 的 VoIP …… 453
9.4 实时会话式应用的协议 …… 455
9.4.1 RTP …… 455
9.4.2 SIP …… 457
9.5 支持多媒体的网络 …… 461
9.5.1 定制尽力而为网络 …… 462
9.5.2 提供多种类型的服务 …… 463
9.5.3 区分服务 …… 468
9.5.4 每连接服务质量保证：资源预约和呼叫准入 …… 470
9.6 小结 …… 472
课后习题和问题 …… 473
复习题 …… 473
习题 …… 473
编程作业 …… 477
人物专访 …… 478
参考文献 …… 480

计算机网络和因特网

今天的因特网无疑是有史以来由人类创造、精心设计的最大系统，该系统具有数以亿计的相连的计算机、通信链路和交换机，有数十亿的用便携计算机、平板电脑和智能手机连接的用户，并且还有一批与因特网连接的“物品”，包括游戏机、监视系统、手表、眼镜、温度调节装置、体重计和汽车。面对如此巨大并且具有如此众多不同组件和用户的因特网，是否能够理解它的工作原理？是否存在某些指导原则和结构，能够作为理解这种规模和复杂程度惊人的系统的基础？这样的话，能让学习计算机网络成为既引人入胜又趣味盎然的事吗？幸运的是，对所有这些问题都是响亮的肯定答复。本书的目的的确是向读者介绍计算机网络这个动态领域的最新知识，使他们深入地理解网络的原则和实践，使读者不仅能理解今天的网络，而且能理解明天的网络。

第1章概述了计算机网络和因特网。这一章的目标是从整体上粗线条地勾勒出计算机网络的概貌，并且描述本书内容的框架。这一章包括大量的背景知识，讨论大量的计算机网络构件，而且将它们放在整个网络的大环境中进行讨论。

本章将以如下方式组织计算机网络的概述：在介绍了某些基本术语和概念后，将首先查看构成网络的基本硬件和软件组件。我们从网络的边缘开始，考察在网络中运行的端系统和网络应用；接下来探究计算机网络的核心，查看传输数据的链路和交换机，以及将端系统与网络核心相连接的接入网和物理媒体。我们将了解因特网是“网络的网络”，并将得知这些网络是怎样彼此连接起来的。

在浏览完计算机网络的边缘和核心之后，本章的后半部分将从更广泛、更抽象的角度来考察计算机网络。我们将考察在计算机网络中数据的时延、丢包和吞吐量，给出一个端到端吞吐量和时延的简单定量模型：该模型兼顾了传输、传播和排队时延等因素。接下来，我们将介绍计算机联网时的一些关键的体系结构原则，如协议分层和服务模型。我们还将了解到计算机网络对于许多不同类型的攻击来说是脆弱的，将回顾其中的某些攻击并且考虑使计算机网络更为安全的方法。最后，我们将以计算机网络的简要历史结束本章的学习。

1.1 什么是因特网

在本书中，我们使用一种特定的计算机网络，即公共因特网，作为讨论计算机网络及其协议的主要载体。但什么是因特网？回答这个问题有两种方式：其一，我们能够描述因特网的具体构成，即构成因特网的基本硬件和软件组件；其二，我们能够根据为分布式应用提供服务的联网基础设施来描述因特网。我们先从描述因特网的具体构成开始，并用图1-1举例说明我们的讨论。

1.1.1 具体构成描述

因特网是一个世界范围的计算机网络，即它是一个互联了遍及全世界数十亿计算设备的网络。在不久前，这些计算设备多数是传统的桌面PC、Linux工作站以及所谓的服务器

（它们用于存储和传输 Web 页面和电子邮件报文等信息）。然而，越来越多的非传统的因特网“物品”（如便携机、智能手机、平板电脑、电视、游戏机、温度调节装置、家用安全系统、家用电器、手表、眼镜、汽车、运输控制系统等）正在与因特网相连。的确，在许多非传统设备连接到因特网的情况下，计算机网络（computer network）这个术语开始听起来有些过时了。用因特网术语来说，所有这些设备都称为**主机**（host）或**端系统**（end system）。据估计，到 2015 年有大约 50 亿台设备与因特网连接，而到 2020 年该数字将达 250 亿［Gartner 2014］。2015 年，全世界已有超过 32 亿因特网用户，接近世界人口的 40%［ITU 2015］。

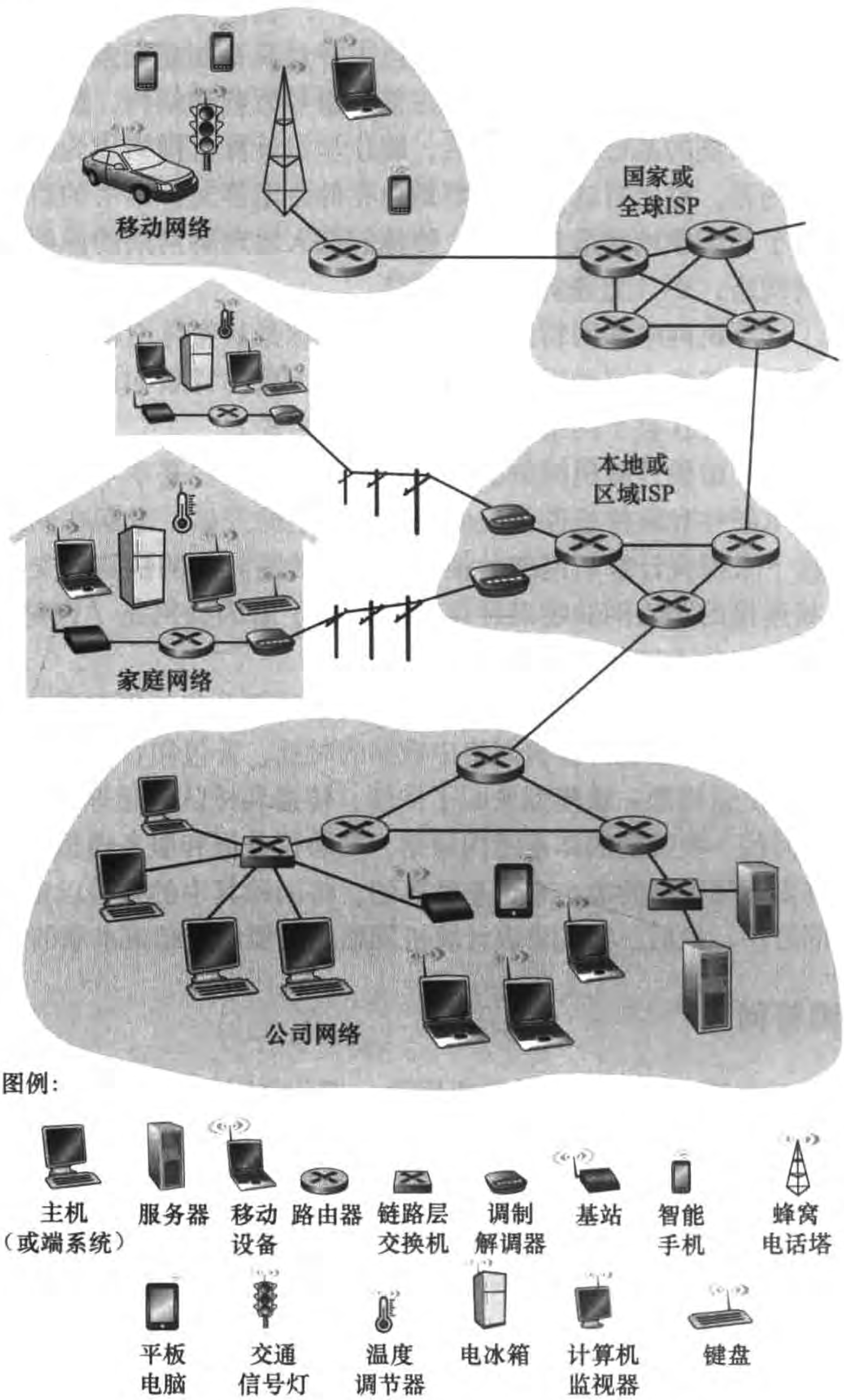


图 1-1 因特网的一些部件

端系统通过**通信链路**（communication link）和**分组交换机**（packet switch）连接到一起。在1.2节中，我们将介绍许多类型的通信链路，它们由不同类型的物理媒体组成。这些物理媒体包括同轴电缆、铜线、光纤和无线电频谱。不同的链路能够以不同的速率传输数据，链路的**传输速率**（transmission rate）以比特/秒（bit/s，或bps）度量。当一台端系统要向另一台端系统发送数据时，发送端系统将数据分段，并为每段加上首部字节。由此形成的信息包用计算机网络的术语来说称为**分组**（packet）。这些分组通过网络发送到目的端系统，在那里被装配成初始数据。

分组交换机从它的一条入通信链路接收到达的分组，并从它的一条出通信链路转发该分组。市面上流行着各种类型、各具特色的分组交换机，但在当今的因特网中，两种最著名的类型是**路由器**（router）和**链路层交换机**（link-layer switch）。这两种类型的交换机朝着最终目的地转发分组。链路层交换机通常用于接入网中，而路由器通常用于网络核心中。从发送端系统到接收端系统，一个分组所经历的一系列通信链路和分组交换机称为通过该网络的**路径**（route或path）。思科公司估计到2016年年底全球年度IP流量将超过泽字节（zettabyte，10^{21}字节），到2019年将达每年2泽字节［Cisco VNI 2015］。

用于传送分组的分组交换网络在许多方面类似于承载运输车辆的运输网络，该网络包括了高速公路、公路和交叉口。例如，考虑下列情况，一个工厂需要将大量货物搬运到数千公里以外的某个目的地仓库。在工厂中，货物要分开并装上卡车车队。然后，每辆卡车独立地通过高速公路、公路和立交桥组成的网络向仓库运送货物。在目的地仓库，卸下这些货物，并且与一起装载的同一批货物的其余部分堆放在一起。因此，在许多方面，分组类似于卡车，通信链路类似于高速公路和公路，分组交换机类似于交叉口，而端系统类似于建筑物。就像卡车选取运输网络的一条路径前行一样，分组则选取计算机网络的一条路径前行。

端系统通过**因特网服务提供商**（Internet Service Provider，ISP）接入因特网，包括如本地电缆或电话公司那样的住宅区ISP、公司ISP、大学ISP，在机场、旅馆、咖啡店和其他公共场所提供WiFi接入的ISP，以及为智能手机和其他设备提供移动接入的蜂窝数据ISP。每个ISP自身就是一个由多台分组交换机和多段通信链路组成的网络。各ISP为端系统提供了各种不同类型的网络接入，包括如线缆调制解调器或DSL那样的住宅宽带接入、高速局域网接入和移动无线接入。ISP也为内容提供者提供因特网接入服务，将Web站点和视频服务器直接连入因特网。因特网就是将端系统彼此互联，因此为端系统提供接入的ISP也必须互联。较低层的ISP通过国家的、国际的较高层ISP（如Level 3 Communications、AT&T、Sprint和NTT）互联起来。较高层ISP是由通过高速光纤链路互联的高速路由器组成的。无论是较高层还是较低层ISP网络，它们每个都是独立管理的，运行着IP协议（详情见后），遵从一定的命名和地址规则。我们将在1.3节中更为详细地考察ISP及其互联的情况。

端系统、分组交换机和其他因特网部件都要运行一系列**协议**（protocol），这些协议控制因特网中信息的接收和发送。TCP（Transmission Control Protocol，传输控制协议）和IP（Internet Protocol，网际协议）是因特网中两个最为重要的协议。IP协议定义了在路由器和端系统之间发送和接收的分组格式。因特网的主要协议统称为TCP/IP。我们在这一章中就开始接触这些协议。但这仅仅是个开始，本书的许多地方与计算机网络协议有关。

鉴于因特网协议的重要性，每个人就各个协议及其作用取得一致认识是很重要的，这样人们就能够创造协同工作的系统和产品。这正是标准发挥作用的地方。**因特网标准**

(Internet standard) 由因特网工程任务组 (Internet Engineering Task Force, IETF) [IETF 2016] 研发。IETF的标准文档称为**请求评论** (Request For Comment, RFC)。RFC最初只是普通的请求评论 (因此而得名), 目的是解决因特网先驱者们面临的网络和协议问题 [Allman 2011]。RFC文档往往是技术性很强并相当详细的。它们定义了TCP、IP、HTTP (用于Web) 和SMTP (用于电子邮件) 等协议。目前已经有将近7000个RFC。其他组织也在制定用于网络组件的标准, 最引人注目的是针对网络链路的标准。例如, IEEE 802 LAN/MAN标准化委员会 [IEEE 802 2016] 制定了以太网和无线WiFi的标准。

1.1.2 服务描述

前面的讨论已经辨识了构成因特网的许多部件。但是我们也能从一个完全不同的角度, 即从*为应用程序提供服务的基础设施*的角度来描述因特网。除了诸如电子邮件和Web冲浪等传统应用外, 因特网应用还包括移动智能手机和平板电脑应用程序, 其中包括即时讯息、与实时道路流量信息的映射、来自云的音乐流、电影和电视流、在线社交网络、视频会议、多人游戏以及基于位置的推荐系统。因为这些应用程序涉及多个相互交换数据的端系统, 故它们被称为**分布式应用程序** (distributed application)。重要的是, 因特网应用程序运行在端系统上, 即它们并不运行在网络核心中的分组交换机中。尽管分组交换机能够加速端系统之间的数据交换, 但它们并不在意作为数据的源或宿的应用程序。

我们稍深入地探讨一下为应用程序提供服务的基础设施的含义。为此, 假定你对某种分布式因特网应用有一个激动人心的新想法, 它可能大大地造福于人类, 或者它可能直接使你名利双收。你将如何把这种想法转换成一种实际的因特网应用呢? 因为应用程序运行在端系统上, 所以你需要编写运行在端系统上的一些软件。例如, 你可能用Java、C或Python编写软件。此时, 因为你在研发一种分布式因特网应用程序, 运行在不同端系统上的软件将需要互相发送数据。此时我们碰到一个核心问题, 这个问题导致了另一种描述因特网的方法, 即将因特网描述为应用程序的平台。运行在一个端系统上的应用程序怎样才能指令因特网向运行在另一个端系统上的软件发送数据呢?

与因特网相连的端系统提供了一个**套接字接口** (socket interface), 该接口规定了运行在一个端系统上的程序请求因特网基础设施向运行在另一个端系统上的特定目的地程序交付数据的方式。因特网套接字接口是一套发送程序必须遵循的规则集合, 因此因特网能够将数据交付给目的地。我们将在第2章详细讨论因特网套接字接口。此时, 我们做一个简单的类比, 在本书中我们将经常使用这个类比。假定Alice使用邮政服务向Bob发一封信。当然, Alice不能只是写了这封信 (相关数据) 然后把该信丢出窗外。相反, 邮政服务要求Alice将信放入一个信封中; 在信封的中间写上Bob的全名、地址和邮政编码; 封上信封; 在信封的右上角贴上邮票; 最后将该信封丢进一个邮局的邮政服务信箱中。因此, 该邮政服务有自己的 "邮政服务接口" 或一套规则, 这是Alice必须遵循的, 这样邮政服务才能将她的信件交付给Bob。同理, 因特网也有一个发送数据的程序必须遵循的套接字接口, 使因特网向接收数据的程序交付数据。

当然, 邮政服务向顾客提供了多种服务, 如特快专递、挂号、普通服务等。同样, 因特网向应用程序提供了多种服务。当你研发一种因特网应用程序时, 也必须为你的应用程序选择其中的一种因特网服务。我们将在第2章中描述因特网服务。

我们已经给出了因特网的两种描述方法: 一种是根据它的硬件和软件组件来描述, 另

一种是根据基础设施向分布式应用程序提供的服务来描述。但是，你也许还是对什么是因特网感到困惑。什么是分组交换和 TCP/IP？什么是路由器？因特网中正在使用什么样的通信链路？什么是分布式应用？一个恒温调节器或人体秤如何与因特网相连？如果你现在还对这些心存疑惑，请不要担心。这本书除了向你介绍因特网的具体构成外，还要介绍支配因特网的工作原理以及它工作的来龙去脉。我们将在后续章节中解释这些重要的术语和问题。

1.1.3 什么是协议

既然我们已经对因特网是什么有了一点印象，那么下面考虑计算机网络中另一个重要的时髦术语：协议（protocol）。什么是协议？协议是用来干什么的？

1. 人类活动的类比

也许理解计算机网络协议这一概念的一个最容易办法是，先与某些人类活动进行类比，因为我们人类无时无刻不在执行协议。考虑当你想要向某人询问时间时将要怎样做。图 1-2 中显示了一种典型的交互过程。人类协议（至少说是好的行为方式）要求一方首先进行问候（图 1-2 中的第一个“你好”），以开始与另一个人的通信。对“你好”的典型响应是返回一个“你好”报文。此人用一个热情的“你好”进行响应，隐含着一种指示，表明能够继续向那人询问时间了。对最初的“你好”的不同响应（例如“不要烦我!”，或“我不会说英语”，或某些不合时宜的回答）也许表明了一个勉强的或不能进行的通信。在此情况下，按照人类协议，发话者就不能询问时间了。有时，问的问题根本得不到任何回答，在此情况下，发话者通常会放弃向这个人询问时间。注意在我们人类协议中，有我们发送的特定报文，也有我们根据接收到的应答报文或其他事件（例如在某个给定的时间内没有回答）采取的动作。显然，发送和接收的报文，以及这些报文发送和接收或其他事件出现时所采取的动作，这些在一个人类协议中起到了核心作用。如果人们使用不同

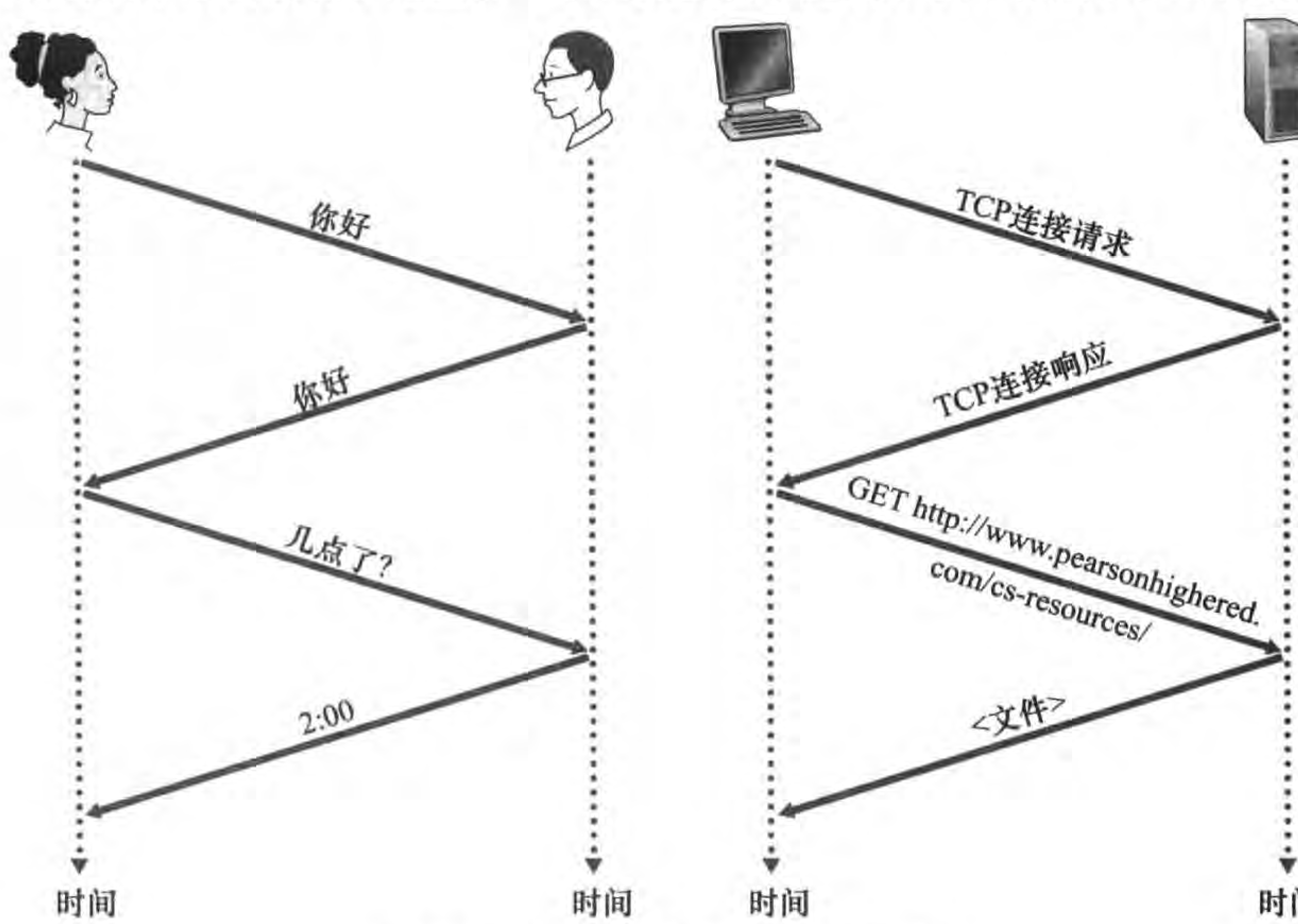


图 1-2 人类协议和计算机网络协议

的协议（例如，如果一个人讲礼貌，而另一人不讲礼貌，或一个人明白时间的概念，而另一人却不理解），这些协议就不能交互，因而不能完成有用的工作。在网络中这个道理同样成立。即为了完成一项工作，要求两个（或多个）通信实体运行相同的协议。

我们再考虑第二个人类类比的例子。假定你正在大学课堂里上课（例如上的是计算机网络课程）。教师正在唠唠叨叨地讲述协议，而你困惑不解。这名教师停下来问："同学们有什么问题吗?"（教师发送出一个报文，该报文被所有没有睡觉的学生接收到了。）你举起了手（向教师发送了一个隐含的报文）。这位教师面带微笑地示意你说："请讲……"（教师发出的这个报文鼓励你提出问题，教师喜欢被问问题。）接着你就问了问题（向该教师传输了你的报文）。教师听取了你的问题（接收了你的问题报文）并加以回答（向你传输了回答报文）。我们再一次看到了报文的发送和接收，以及这些报文发送和接收时所采取的一系列约定俗成的动作，这些是这个"提问与回答"协议的关键所在。

2. 网络协议

网络协议类似于人类协议，除了交换报文和采取动作的实体是某些设备（可以是计算机、智能手机、平板电脑、路由器或其他具有网络能力的设备）的硬件或软件组件。在因特网中，涉及两个或多个远程通信实体的所有活动都受协议的制约。例如，在两台物理上连接的计算机中，硬件实现的协议控制了在两块网络接口卡间的"线上"的比特流；在端系统中，拥塞控制协议控制了在发送方和接收方之间传输的分组发送的速率；路由器中的协议决定了分组从源到目的地的路径。在因特网中协议运行无处不在，因此本书的大量篇幅都与计算机网络协议有关。

以大家可能熟悉的一个计算机网络协议为例，考虑当你向一个 Web 服务器发出请求（即你在 Web 浏览器中键入一个 Web 网页的 URL）时所发生的情况。图 1-2 右半部分显示了这种情形。首先，你的计算机将向该 Web 服务器发送一条连接请求报文，并等待回答。该 Web 服务器将最终能接收到连接请求报文，并返回一条连接响应报文。得知请求该 Web 文档正常以后，计算机则在一条 GET 报文中发送要从这台 Web 服务器上取回的网页名字。最后，Web 服务器向计算机返回该 Web 网页（文件）。

从上述的人类活动和网络例子中可见，报文的交换以及发送和接收这些报文时所采取的动作是定义一个协议的关键元素：

> **协议**（protocol）*定义了在两个或多个通信实体之间交换的报文的格式和顺序，以及报文发送和/或接收一条报文或其他事件所采取的动作。*

因特网（更一般地说是计算机网络）广泛地使用了协议。不同的协议用于完成不同的通信任务。当你阅读完这本书后将会知道，某些协议简单而直截了当，而某些协议则复杂且晦涩难懂。掌握计算机网络领域知识的过程就是理解网络协议的构成、原理和工作方式的过程。

1.2　网络边缘

在上一节中，我们给出了因特网和网络协议的总体概述。现在我们将更深入一些来探究计算机网络（特别是因特网）的部件。在本节中，我们从网络边缘开始，观察一下我们更为熟悉的部件，即我们日常使用的计算机、智能手机和其他设备。在接下来的一节中，我们将从网络边缘向网络核心推进，查看计算机网络中的交换和选路。

回想前一节中计算机网络的术语，通常把与因特网相连的计算机和其他设备称为端系统。如图1-3所示，因为它们位于因特网的边缘，故而被称为端系统。因特网的端系统包括了桌面计算机（例如，桌面PC、Mac和Linux设备）、服务器（例如，Web和电子邮件服务器）和移动计算机（例如，便携机、智能手机和平板电脑）。此外，越来越多的非传统物品正被作为端系统与因特网相连（参见“历史事件”）。

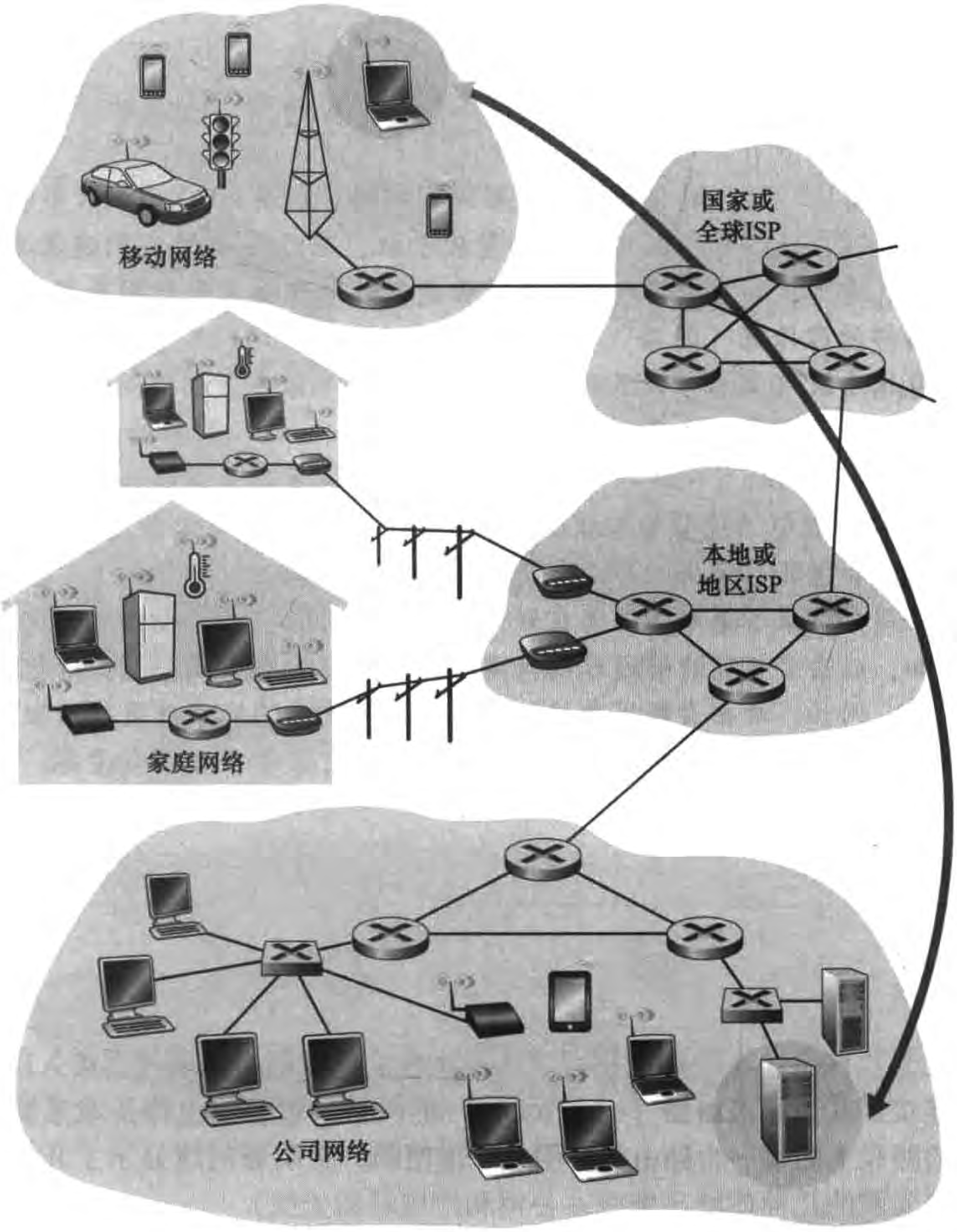


图1-3 端系统交互

端系统也称为主机（host），因为它们容纳（即运行）应用程序，如Web浏览器程序、Web服务器程序、电子邮件客户程序或电子邮件服务器程序等。本书通篇将交替使用主机和端系统这两个术语，即*主机=端系统*。主机有时又被进一步划分为两类：**客户**（client）和**服务器**（server）。客户通常是桌面PC、移动PC和智能手机等，而服务器通常是更为强大的机器，用于存储和发布Web页面、流视频、中继电子邮件等。今天，大部分提供搜索结果、电子邮件、Web页面和视频的服务器都属于大型**数据中心**（data center）。例如，谷歌公司（Google）拥有50～100个数据中心，其中15个大型数据中

心每个都有 10 万台以上的服务器。

历史事件

物联网

你能够想象几乎每样东西都与因特网无线连接的一个世界吗?能够想象在这个世界上连接了大多数人、汽车、自行车、眼镜、手表、玩具、医疗设备、家用传感器、教室、视频监视系统、大气传感器、陈列架产品和宠物吗?这种物联网(Internet of Things, IoT)事实上就在不远的某处。

据估计,在 2015 年有 50 亿件物品连接到因特网,到了 2020 年该数字能够达到 250 亿[Gartner 2014]。这些物品包括我们的智能手机,它已经跟随我们遍及我们的家庭、办公室和汽车,向 ISP 和因特网应用程序报告我们的地理位置和使用数据。而除了智能手机,各种各样的非传统"物品"也已经作为产品可供使用。例如,有因特网连接的可穿戴设备,包括手表(来自苹果和许多其他生产商)和眼镜。因特网连接的眼镜能够将我们看到的每样东西上载到云端,允许我们与世界各地的人们实时共享我们的视觉体验。对于智能家居,有因特网连接的物品,包括恒温调节器(能够通过智能手机远程控制)以及人体秤(能够通过智能手机以图形化方式审查我们的饮食)。有因特网连接的玩具,包括识别和解释孩子讲话并适当地进行响应的玩偶。

物联网给用户带来了革命性的潜在好处。但与此同时,也存在巨大的安全风险和隐私风险。例如,攻击者可能能够通过因特网入侵物联网设备或服务器,以收集来自物联网设备的数据。例如,攻击者能够劫持一个与因特网连接的玩偶并直接与孩子交谈,或者能够入侵存储个人健康或活动信息的数据库,这些信息来自可穿戴设备。这些安全和隐私问题可能会使消费者渐渐地失去信心,并可能导致消费者的广泛接受程度下降[FTC 2015]。

1.2.1 接入网

考虑了位于"网络边缘"的应用程序和端系统后,我们接下来考虑**接入网**,这是指将端系统物理连接到其**边缘路由器**(edge router)的网络。边缘路由器是端系统到任何其他远程端系统的路径上的第一台路由器。图 1-4 用粗的、带阴影的线显示了几种类型的接入链路和使用接入网的几种环境(家庭、公司和广域移动无线)。

1. 家庭接入:DSL、电缆、FTTH、拨号和卫星

2014 年,在发达国家中,实现因特网接入的家庭超过 78%,名列前茅的韩国、荷兰、芬兰和瑞典实现因特网接入的家庭则超过 80%,几乎所有家庭都是经过高速宽带连接[ITU 2015]。考虑到家庭对网络接入的强烈兴趣,我们从家庭与因特网的连接开始介绍接入网概况。

今天,宽带住宅接入有两种最流行的类型:**数字用户线**(Digital Subscriber Line, DSL)和电缆。住户通常从提供本地电话接入的本地电话公司处获得 DSL 因特网接入。因此,当使用 DSL 时,用户的本地电话公司也是它的 ISP。如图 1-5 所示,每个用户的 DSL 调制解调器使用现有的电话线(即双绞铜线,将在 1.2.2 节中讨论它)与位于电话公司的

本地中心局（CO）中的数字用户线接入复用器（DSLAM）交换数据。家庭的DSL调制解调器得到数字数据后将其转换为高频音，以通过电话线传输给本地中心局；来自许多家庭的模拟信号在DSLAM处被转换回数字形式。

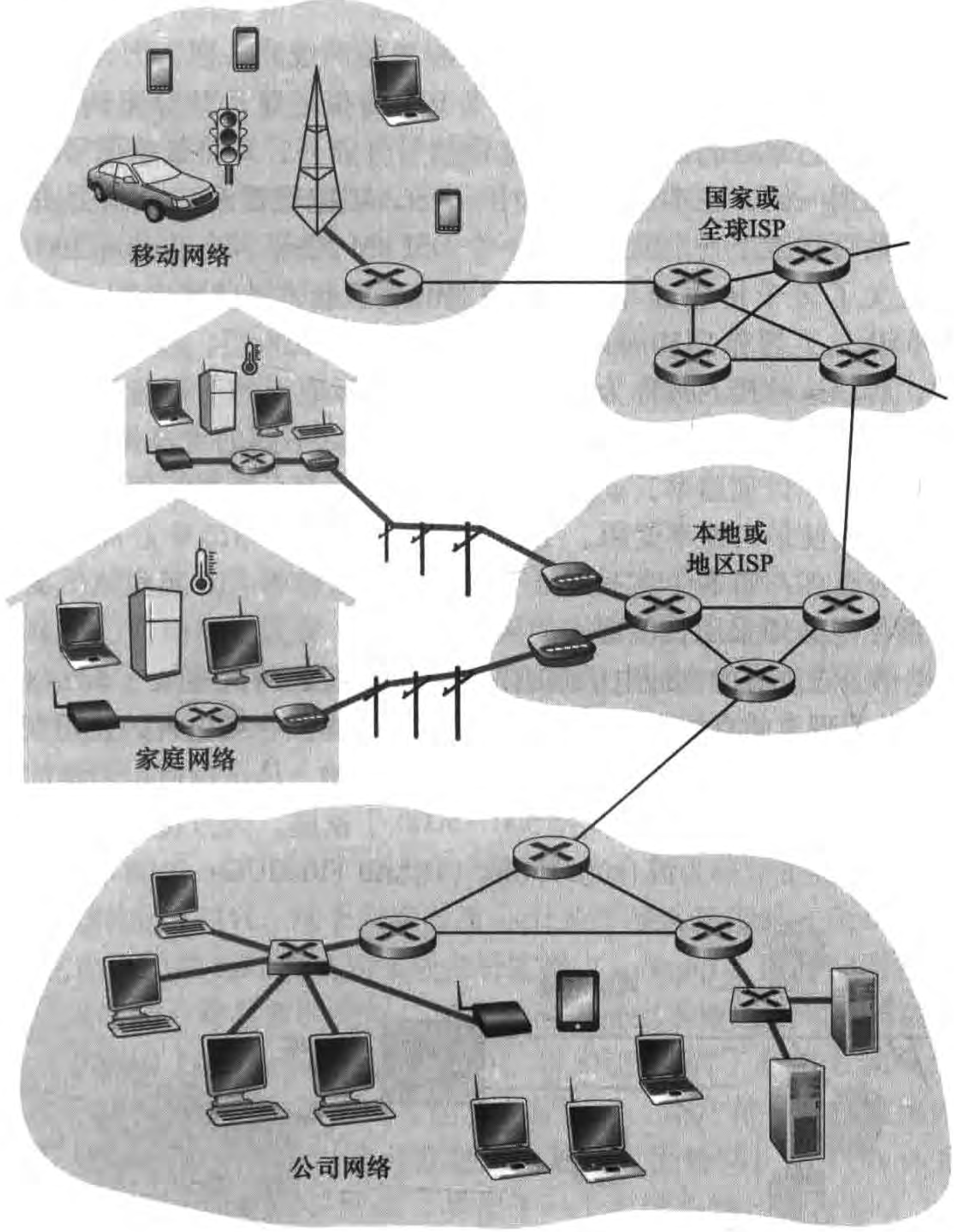


图1-4 接入网

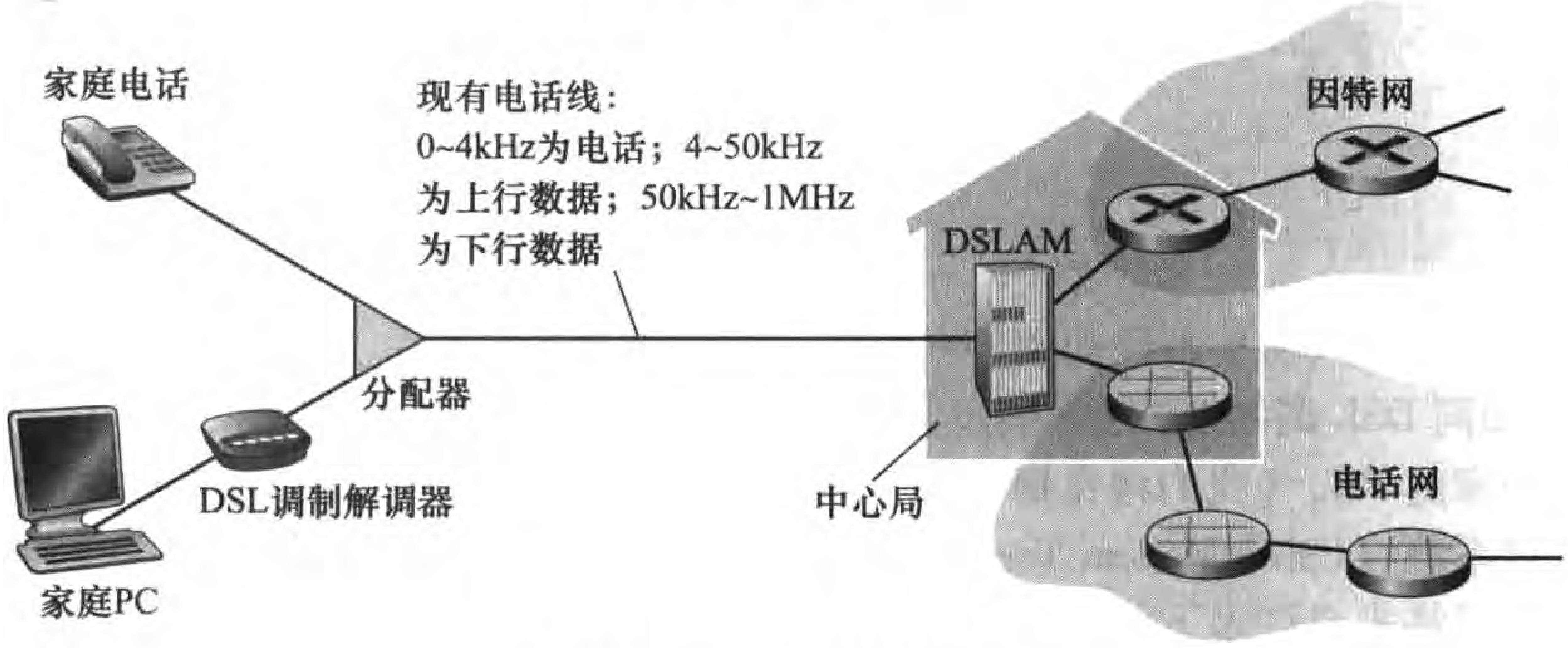


图1-5 DSL因特网接入

家庭电话线同时承载了数据和传统的电话信号，它们用不同的频率进行编码：

- 高速下行信道，位于 50kHz 到 1MHz 频段；
- 中速上行信道，位于 4kHz 到 50kHz 频段；
- 普通的双向电话信道，位于 0 到 4kHz 频段。

这种方法使单根 DSL 线路看起来就像有 3 根单独的线路一样，因此一个电话呼叫和一个因特网连接能够同时共享 DSL 链路。（1.3.1 节将描述这种频分复用技术。）在用户一侧，一个分配器把到达家庭的数据信号和电话信号分隔开，并将数据信号转发给 DSL 调制解调器。在电话公司一侧，在本地中心局中，DSLAM 把数据和电话信号分隔开，并将数据送往因特网。数百甚至上千个家庭与同一个 DSLAM 相连［Dischinger 2007］。

DSL 标准定义了多个传输速率，包括 12Mbps 下行和 1.8Mbps 上行传输速率［ITU 1999］，以及 55Mbps 下行和 15Mbps 上行传输速率［ITU 2006］。因为这些上行速率和下行速率是不同的，所以这种接入被称为是不对称的。实际取得的下行和上行传输速率也许小于上述速率，因为当 DSL 提供商提供分等级的服务（以不同的价格使用不同的速率）时，他们也许有意地限制了住宅速率，或者因为家庭与本地中心局之间的距离、双绞线的规格和电气干扰的程度而使最大速率受限。工程师特别为家庭与本地中心局之间的短距离接入设计了 DSL；一般而言，如果住宅不是位于本地中心局的 5～10 英里（1 英里 =1609.344 米）范围内，该住宅必须采用其他形式的因特网接入。

DSL 利用电话公司现有的本地电话基础设施，而**电缆因特网接入**（cable Internet access）利用了有线电视公司现有的有线电视基础设施。住宅从提供有线电视的公司获得了电缆因特网接入。如图 1-6 所示，光缆将电缆头端连接到地区枢纽，从这里使用传统的同轴电缆到达各家各户和公寓。每个地区枢纽通常支持 500～5000 个家庭。因为在这个系统中应用了光纤和同轴电缆，所以它经常被称为混合光纤同轴（Hybrid Fiber Coax，HFC）系统。

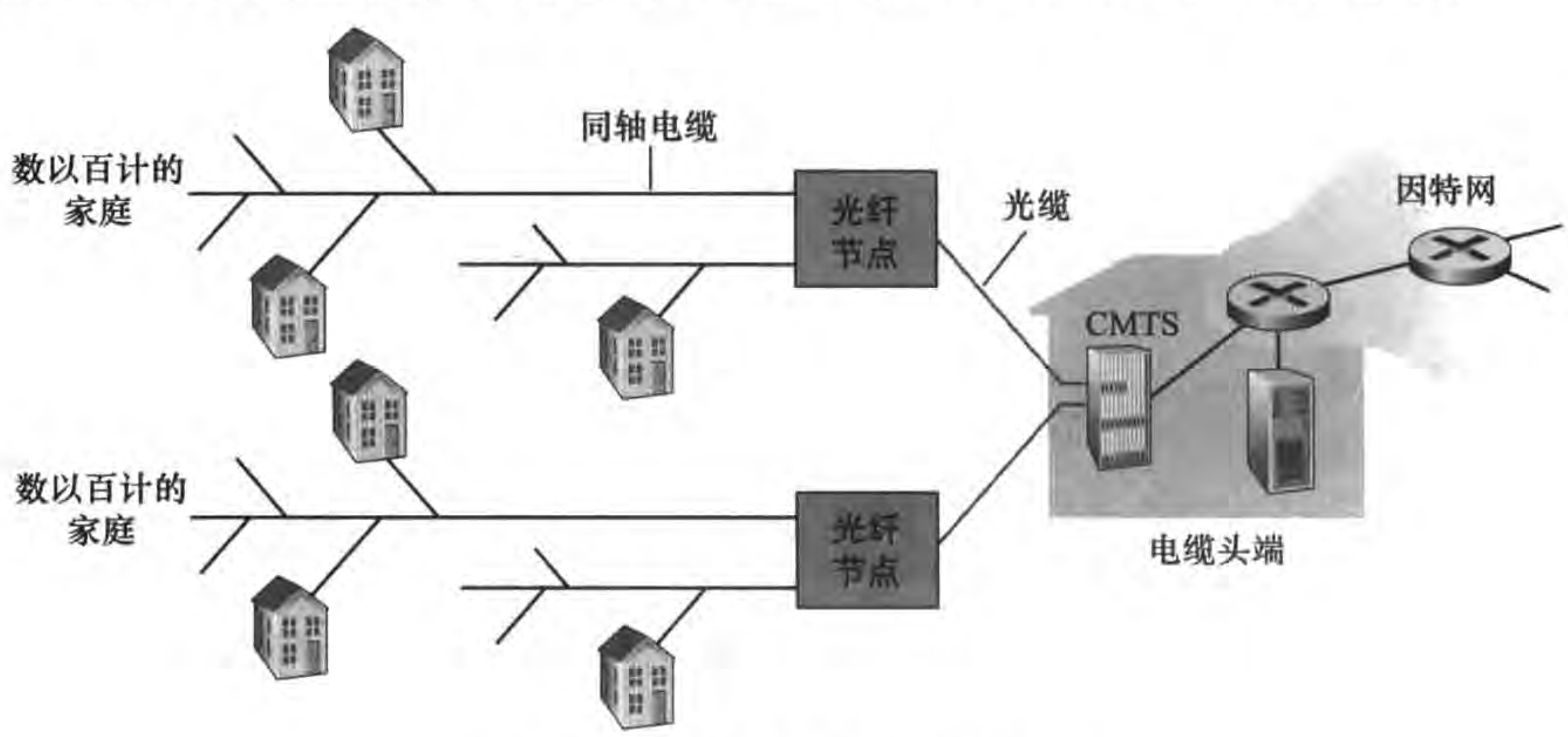


图 1-6 一个混合光纤同轴接入网

电缆因特网接入需要特殊的调制解调器，这种调制解调器称为电缆调制解调器（cable modem）。如同 DSL 调制解调器，电缆调制解调器通常是一个外部设备，通过一个以太网端口连接到家庭 PC。（我们将在第 6 章非常详细地讨论以太网。）在电缆头端，电缆调制解调器端接系统（Cable Modem Termination System，CMTS）与 DSL 网络的 DSLAM 具有类似的功能，即将来自许多下行家庭中的电缆调制解调器发送的模拟信号转换回数字形式。电缆调制解调器将 HFC 网络划分为下行和上行两个信道。如同 DSL，接入通常是不对称

的，下行信道分配的传输速率通常比上行信道的高。DOCSIS 2.0 标准定义了高达 42.8Mbps 的下行速率和高达 30.7Mbps 的上行速率。如在 DSL 网络中的情况那样，由于较低的合同数据率或媒体损耗，可能不一定能达到最大可取得的速率。

电缆因特网接入的一个重要特征是共享广播媒体。特别是，由头端发送的每个分组向下行经每段链路到每个家庭；每个家庭发送的每个分组经上行信道向头端传输。因此，如果几个用户同时经下行信道下载一个视频文件，每个用户接收视频文件的实际速率将大大低于电缆总计的下行速率。而另一方面，如果仅有很少的活跃用户在进行 Web 冲浪，则每个用户都可以以全部的下行速率接收 Web 网页，因为用户们很少在完全相同的时刻请求网页。因为上行信道也是共享的，需要一个分布式多路访问协议来协调传输和避免碰撞。（我们将在第 6 章中更为详细地讨论碰撞问题。）

尽管 DSL 和电缆网络当前代表了超过 85% 的美国住宅宽带接入，但一种提供更高速率的新兴技术是**光纤到户**（Fiber To The Home，FTTH）[FTTH Council 2016]。顾名思义，FTTH 概念简单，从本地中心局直接到家庭提供了一条光纤路径。现在，许多国家或地区，包括阿拉伯联合酋长国、韩国、中国香港、日本、新加坡、中国台湾、立陶宛和瑞典，入户率已经超过了 30% [FTTH Council 2016]。

从本地中心局到家庭有几种有竞争性的光纤分布方案。最简单的光纤分布网络称为直接光纤，从本地中心局到每户设置一根光纤。更为一般的是，从中心局出来的每根光纤实际上由许多家庭共享，直到相对接近这些家庭的位置，该光纤才分成每户一根光纤。进行这种分配有两种有竞争性的光纤分布体系结构：主动光纤网络（Active Optical Network，AON）和被动光纤网络（Passive Optical Network，PON）。AON 本质上就是交换因特网，我们将在第 6 章讨论它。

这里，我们简要讨论一下 PON，该技术用于 Verizon 的 FIOS 服务中。图 1-7 显示了使用 PON 分布体系结构的 FTTH。每个家庭具有一个光纤网络端接器（Optical Network Terminator，ONT），它由专门的光纤连接到邻近的分配器（splitter）。该分配器把一些家庭（通常少于 100 个）集结到一根共享的光纤，该光纤再连接到本地电话和公司的中心局中的光纤线路端接器（Optical Line Terminator，OLT）。该 OLT 提供了光信号和电信号之间的转换，经过本地电话公司路由器与因特网相连。在家庭中，用户将一台家庭路由器（通常是无线路由器）与 ONT 相连，并经过这台家庭路由器接入因特网。在 PON 体系结构中，所有从 OLT 发送到分配器的分组在分配器（类似于一个电缆头端）处复制。

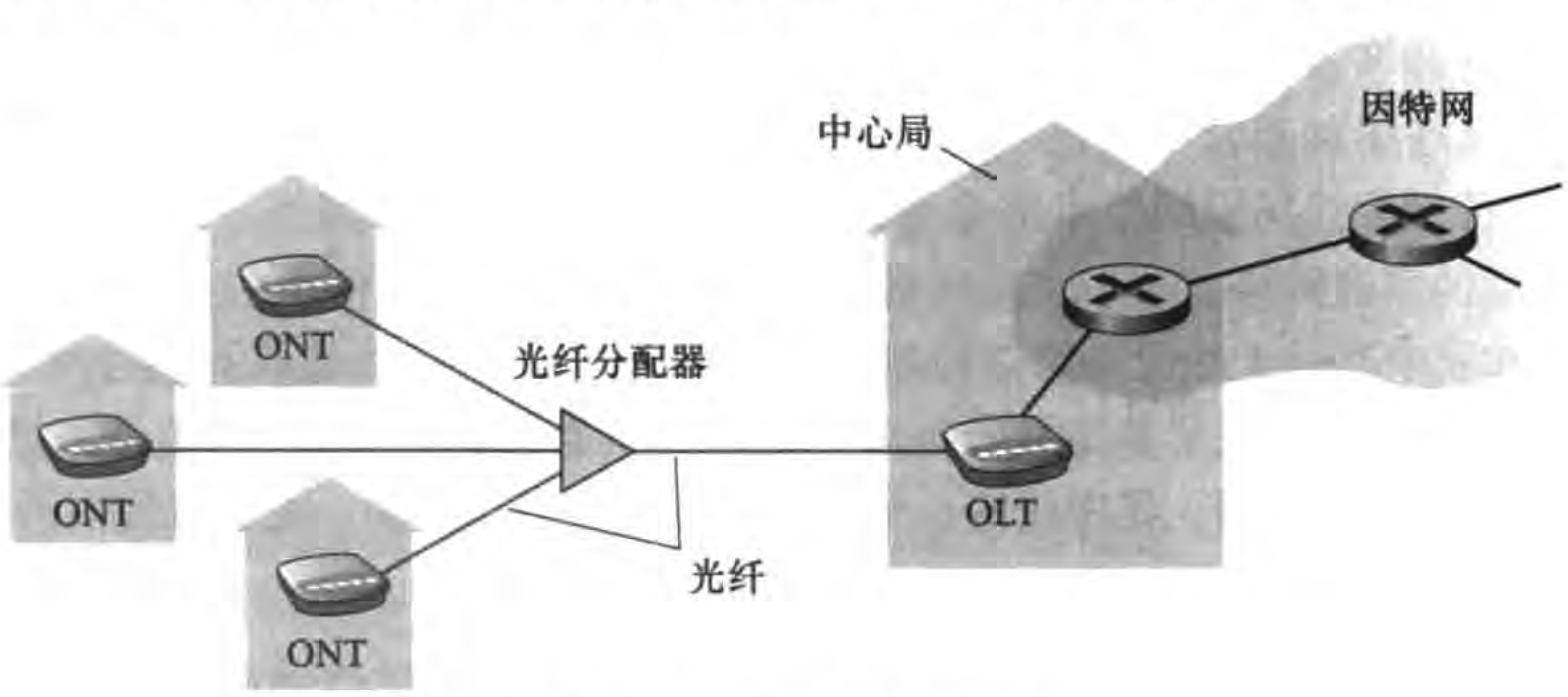


图 1-7 FTTH 因特网接入

FTTH 有潜力提供每秒千兆比特范围的因特网接入速率。然而，大多数 FTTH ISP 提供

多种不同速率选择，较高的速率自然花费更多。美国 FTTH 用户在 2011 年的平均下行速率大约为 20Mbps（与 13Mbps 电缆接入网相当，而小于 5Mbps DSL）［FTTH Council 2011b］。

还可采用另外两种接入网技术为家庭提供因特网接入。在无法提供 DSL、电缆和 FTTH 的地方（例如在某些乡村环境），能够使用卫星链路将住宅以超过 1Mbps 的速率与因特网相连。StarBand 和 HughesNet 是两家这样的卫星接入提供商。使用传统电话线的拨号接入与 DSL 基于相同的模式：家庭的调制解调器经过电话线连接到 ISP 的调制解调器。与 DSL 和其他宽带接入网相比，拨号接入 56kbps 的慢速率是令人痛苦的。

2. 企业（和家庭）接入：以太网和 WiFi

在公司和大学校园以及越来越多的家庭环境中，使用局域网（LAN）将端系统连接到边缘路由器。尽管有许多不同类型的局域网技术，但是以太网到目前为止是公司、大学和家庭网络中最为流行的接入技术。如图 1-8 中所示，以太网用户使用双绞铜线与一台以太网交换机相连，第 6 章中将详细讨论该技术。以太网交换机或这样相连的交换机网络，则再与更大的因特网相连。使用以太网接入，用户通常以 100Mbps 或 1Gbps 速率接入以太网交换机，而服务器可能具有 1Gbps 甚至 10Gbps 的接入速率。

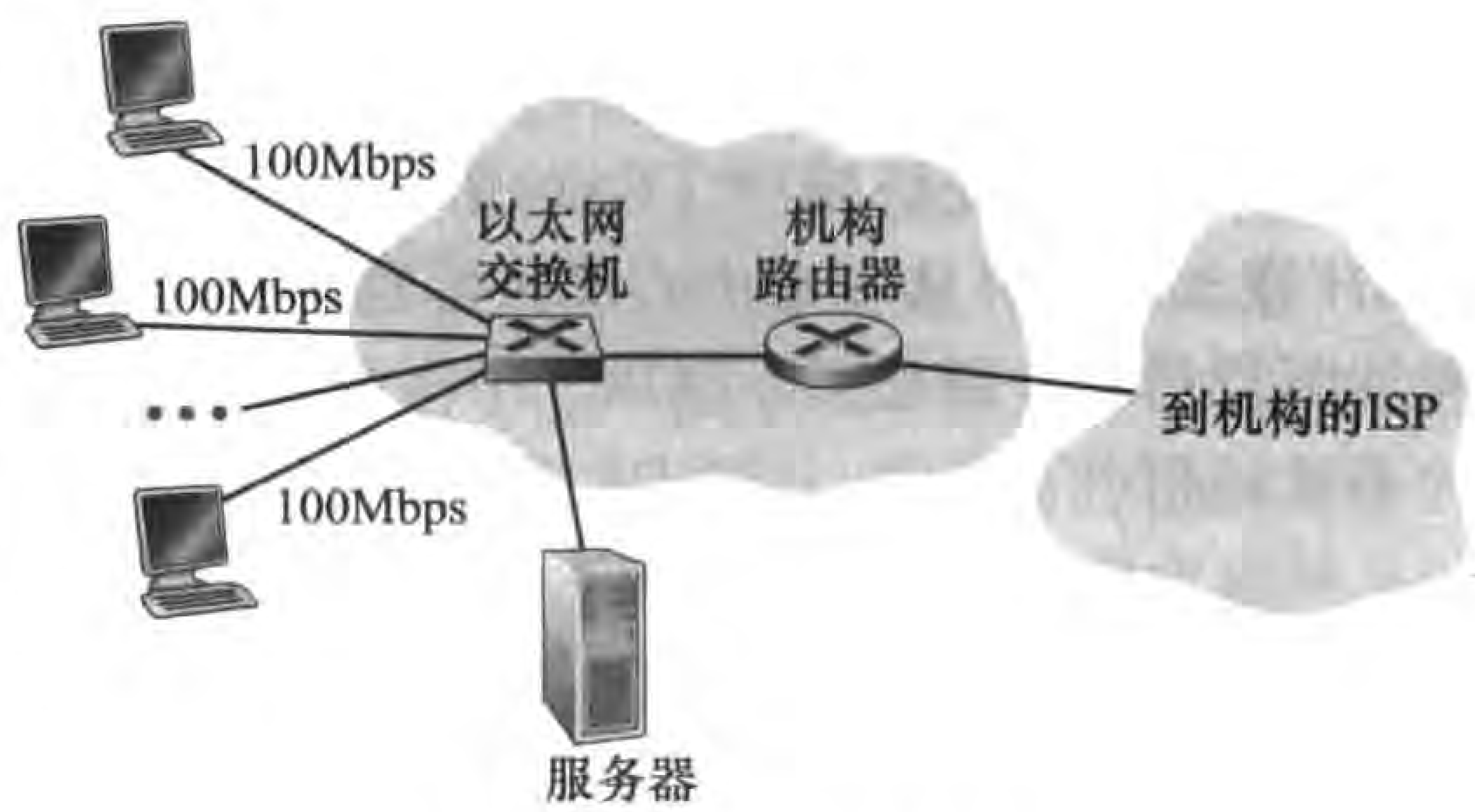


图 1-8　以太网因特网接入

然而，越来越多的人从便携机、智能手机、平板电脑和其他物品无线接入因特网（参见前面的插入内容“物联网”）。在无线 LAN 环境中，无线用户从/到一个接入点发送/接收分组，该接入点与企业网连接（很可能使用了有线以太网），企业网再与有线因特网相连。一个无线 LAN 用户通常必须位于接入点的几十米范围内。基于 IEEE 802.11 技术的无线 LAN 接入，更通俗地称为 WiFi，目前几乎无所不在，如大学、商业办公室、咖啡厅、机场、家庭，甚至在飞机上。在许多城市，人们能够站在街头而位于 10 或 20 个基站范围内（对于可浏览的 802.11 基站全局图，参见［wigle.net 2016］，这些基站由那些非常热心做这种事情的人发现并记载在 Web 站点上）。如在第 7 章详细讨论的那样，802.11 今天提供了高达 100Mbps 的共享传输速率。

虽然以太网和 WiFi 接入网最初是设置在企业（公司或大学）环境中的，但它们近来已经成为家庭网络中相当常见的部件。今天许多家庭将宽带住宅接入（即电缆调制解调器或 DSL）与廉价的无线局域网技术结合起来，以产生强大的家用网络［Edwards 2011］。图 1-9 显示了典型的家庭网络。这个家庭网络组成如下：一台漫游的便携机和一台有线 PC；一个与无线 PC 和家中其他无线设备通信的基站（无线接入点）；一个提供与因特网宽带接入的电缆调制解调器；一台互联了基站及带有电缆调制解调器的固定 PC 的路由器。该网络允许家庭成员经宽带接入因特网，其中任何一个家庭成员都可以在厨房、院子或卧室漫游上网。

3. 广域无线接入：3G 和 LTE

iPhone 和安卓等设备越来越多地用来在移动中发信息、在社交网络中分享照片、观看视频和放音乐。这些设备应用了与蜂窝移动电话相同的无线基础设施，通过蜂窝网提供商运营的基站来发送和接收分组。与 WiFi 不同的是，一个用户仅需要位于基站的数万米

（而不是几十米）范围内。

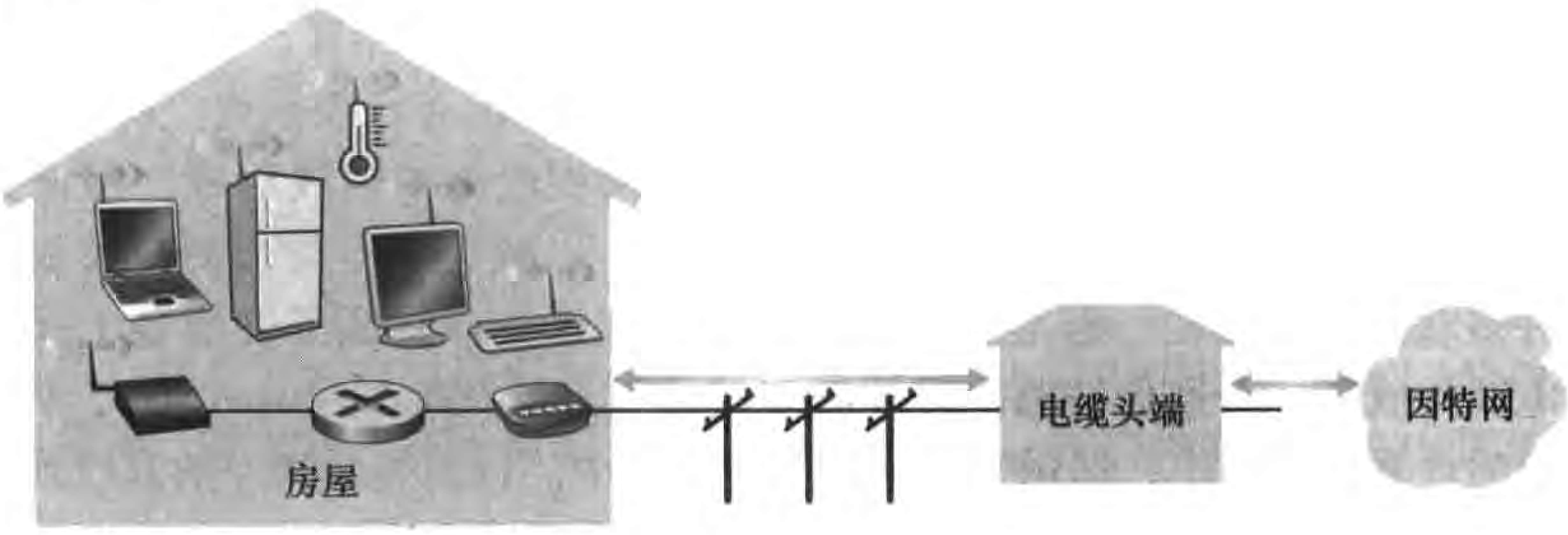


图 1-9 一个典型的家庭网络的示意图

电信公司已经在所谓第三代（3G）无线技术中进行了大量投资，3G 为分组交换广域无线因特网接入提供了超过 1Mbps 的速率。甚至更高速率的广域接入技术即第四代（4G）广域无线网络也已经被部署了。LTE（长期演进“Long-Term Evolution”的缩写，被评为最差首字母缩写词年度奖候选者）来源于 3G 技术，它能够取得超过 10Mbps 的速率。据报道，几十 Mbps 的 LTE 下行速率已经在商业部署中得到应用。我们将在第 7 章讨论无线网络和移动性，以及 WiFi、3G 和 LTE 等技术的基本原则。

1.2.2 物理媒体

在前面的内容中，我们概述了因特网中某些最为重要的网络接入技术。当我们描述这些技术时，我们也指出了所使用的物理媒体。例如，我们说过 HFC 使用了光缆和同轴电缆相结合的技术。我们说过 DSL 和以太网使用了双绞铜线。我们也说过移动接入网使用了无线电频谱。在这一节中，我们简要概述一下这些和其他常在因特网中使用的传输媒体。

为了定义物理媒体所表示的内容，我们仔细思考一下一个比特的短暂历程。考虑一个比特从一个端系统开始传输，通过一系列链路和路由器，到达另一个端系统。这个比特被漫不经心地传输了许许多多次！源端系统首先发射这个比特，不久后其中的第一台路由器接收该比特；第一台路由器发射该比特，接着不久后第二台路由器接收该比特；等等。因此，这个比特当从源到目的地传输时，通过一系列“发射器-接收器”对。对于每个发射器-接收器对，通过跨越一种**物理媒体**（physical medium）传播电磁波或光脉冲来发送该比特。该物理媒体可具有多种形状和形式，并且对沿途的每个发射器-接收器对而言不必具有相同的类型。物理媒体的例子包括双绞铜线、同轴电缆、多模光纤缆、陆地无线电频谱和卫星无线电频谱。物理媒体分成两种类型：**导引型媒体**（guided media）和**非导引型媒体**（unguided media）。对于导引型媒体，电波沿着固体媒体前行，如光缆、双绞铜线或同轴电缆。对于非导引型媒体，电波在空气或外层空间中传播，例如在无线局域网或数字卫星频道中。

在深入讨论各种媒体类型的特性之前，我们简要地讨论一下它们的成本。物理链路（铜线、光缆等）的实际成本与其他网络成本相比通常是相当小的。特别是安装物理链路的劳动力成本能够比材料的成本高出几个数量级。正因为这个原因，许多建筑商在一个建筑物的每个房间中安装了双绞线、光缆和同轴电缆。即使最初仅使用了一种媒体，在不久的将来也可能会使用另一种媒体，这样将来不必再铺设另外的线缆，从而节省了经费。

1. 双绞铜线

最便宜并且最常用的导引型传输媒体是双绞铜线。一百多年来，它一直用于电话网。事实上，从电话机到本地电话交换机的连线超过99%使用的是双绞铜线。我们多数人在自己家中和工作环境中已经看到过双绞线。双绞线由两根绝缘的铜线组成，每根大约1mm粗，以规则的螺旋状排列着。这两根线被绞合起来，以减少邻近类似的双绞线的电气干扰。通常许多双绞线捆扎在一起形成一根电缆，并在这些双绞线外面覆盖上保护性防护层。一对电线构成了一个通信链路。**无屏蔽双绞线**（Unshielded Twisted Pair，UTP）常用在建筑物内的计算机网络中，即用于局域网（LAN）中。目前局域网中的双绞线的数据速率从10Mbps到10Gbps。所能达到的数据传输速率取决于线的粗细以及传输方和接收方之间的距离。

20世纪80年代出现光纤技术时，许多人因为双绞线比特速率低而轻视它，某些人甚至认为光纤技术将完全代替双绞线。但双绞线不是那么容易被抛弃的。现代的双绞线技术例如6a类电缆能够达到10Gbps的数据传输速率，距离长达100m。双绞线最终已经作为高速LAN联网的主导性解决方案。

如前面讨论的那样，双绞线也经常用于住宅因特网接入。我们看到，拨号调制解调器技术通过双绞线能以高达56kbps的速率接入。我们也看到，数字用户线（DSL）技术通过双绞线使住宅用户以超过数十Mbps的速率接入因特网（当用户靠近ISP的中心局居住时）。

2. 同轴电缆

与双绞线类似，同轴电缆由两个铜导体组成，但是这两个导体是同心的而不是并行的。借助于这种结构及特殊的绝缘体和保护层，同轴电缆能够达到较高的数据传输速率。同轴电缆在电缆电视系统中相当普遍。我们前面已经看到，电缆电视系统最近与电缆调制解调器结合起来，为住宅用户提供数十Mbps速率的因特网接入。在电缆电视和电缆因特网接入中，发送设备将数字信号调制到某个特定的频段，产生的模拟信号从发送设备传送到一个或多个接收方。同轴电缆能被用作导引型**共享媒体**（shared medium）。特别是，许多端系统能够直接与该电缆相连，每个端系统都能接收由其他端系统发送的内容。

3. 光纤

光纤是一种细而柔软的、能够导引光脉冲的媒体，每个脉冲表示一个比特。一根光纤能够支持极高的比特速率，高达数十甚至数百Gbps。它们不受电磁干扰，长达100km的光缆信号衰减极低，并且很难窃听。这些特征使得光纤成为长途导引型传输媒体，特别是跨海链路。在美国和别的地方，许多长途电话网络现在全面使用光纤。光纤也广泛用于因特网的主干。然而，高成本的光设备，如发射器、接收器和交换机，阻碍光纤在短途传输中的应用，如在LAN或家庭接入网中就不使用它们。光载波（Optical Carrier，OC）标准链路速率的范围从51.8Mbps到39.8Gbps；这些标准常被称为OC-n，其中的链路速率等于$n \times 51.8$Mbps。目前正在使用的标准包括OC-1、OC-3、OC-12、OC-24、OC-48、OC-96、OC-192、OC-768。[Mukherjee 2006；Ramaswami 2010] 提供了光纤网络各方面的知识。

4. 陆地无线电信道

无线电信道承载电磁频谱中的信号。它不需要安装物理线路，并具有穿透墙壁、提供

与移动用户的连接以及长距离承载信号的能力，因而成为一种有吸引力的媒体。无线电信道的特性极大地依赖于传播环境和信号传输的距离。环境上的考虑取决于路径损耗和遮挡衰落（即当信号跨距离传播和绕过/通过阻碍物体时信号强度降低）、多径衰落（由于干扰对象的信号反射）以及干扰（由于其他传输或电磁信号）。

陆地无线电信道能够大致划分为三类：一类运行在很短距离（如 1 米或 2 米）；另一类运行在局域，通常跨越数十到几百米；第三类运行在广域，跨越数万米。个人设备如无线头戴式耳机、键盘和医疗设备跨短距离运行；在 1.2.1 节中描述的无线 LAN 技术使用了局域无线电信道；蜂窝接入技术使用了广域无线电信道。我们将在第 7 章中详细讨论无线电信道。

5. 卫星无线电信道

一颗通信卫星连接地球上的两个或多个微波发射器/接收器，它们被称为地面站。该卫星在一个频段上接收传输，使用一个转发器（下面讨论）再生信号，并在另一个频率上发射信号。通信中常使用两类卫星：**同步卫星**（geostationary satellite）和**近地轨道**（Low-Earth Orbiting，LEO）卫星［Wiki Satellite 2016］。

同步卫星永久地停留在地球上方的相同点上。这种静止性是通过将卫星置于地球表面上方 36 000km 的轨道上而取得的。从地面站到卫星再回到地面站的巨大距离引入了可观的 280ms 信号传播时延。不过，能以数百 Mbps 速率运行的卫星链路通常用于那些无法使用 DSL 或电缆因特网接入的区域。

近地轨道卫星放置得非常靠近地球，并且不是永久地停留在地球上方的一个点。它们围绕地球旋转，就像月亮围绕地球旋转那样，并且彼此之间可进行通信，也可以与地面站通信。为了提供对一个区域的连续覆盖，需要在轨道上放置许多卫星。当前有许多低轨道通信系统在研制中。LEO 卫星技术未来也许能够用于因特网接入。

1.3 网络核心

在考察了因特网边缘后，我们现在更深入地研究网络核心，即由互联因特网端系统的分组交换机和链路构成的网状网络。图 1-10 用加粗阴影线勾画出网络核心部分。

1.3.1 分组交换

在各种网络应用中，端系统彼此交换**报文**（message）。报文能够包含协议设计者需要的任何东西。报文可以执行一种控制功能（例如，图 1-2 所示例子中的“你好”报文），也可以包含数据，例如电子邮件数据、JPEG 图像或 MP3 音频文件。为了从源端系统向目的端系统发送一个报文，源将长报文划分为较小的数据块，称之为**分组**（packet）。在源和目的地之间，每个分组都通过通信链路和**分组交换机**（packet switch）传送。（交换机主要有两类：**路由器**（router）和**链路层交换机**（link-layer switch）。）分组以等于该链路最大传输速率的速度传输通过通信链路。因此，如果某源端系统或分组交换机经过一条链路发送一个 L 比特的分组，链路的传输速率为 R 比特/秒，则传输该分组的时间为 L/R 秒。

1. 存储转发传输

多数分组交换机在链路的输入端使用**存储转发传输**（store-and-forward transmission）机制。存储转发传输是指在交换机能够开始向输出链路传输该分组的第一个比特之前，必

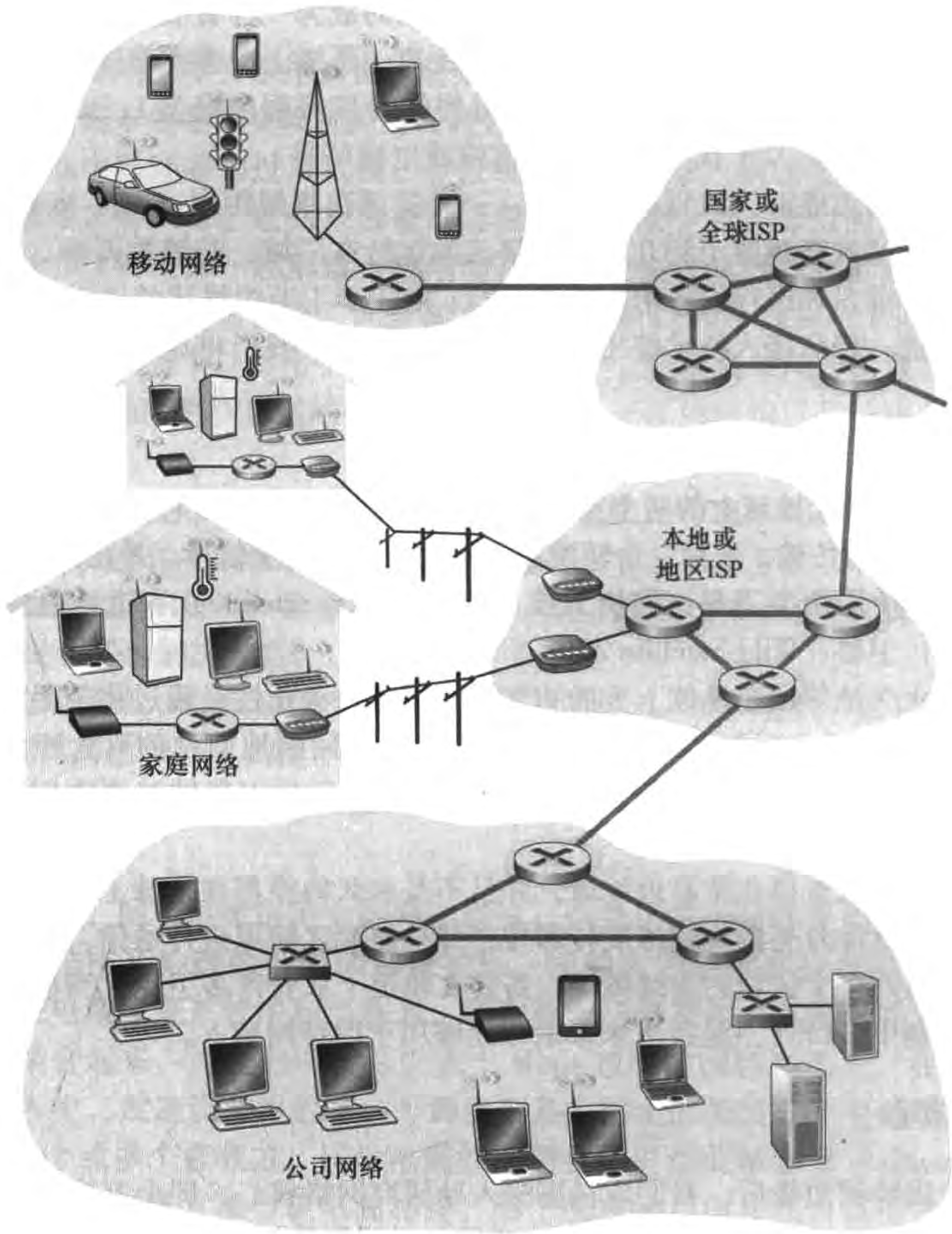


图1-10　网络核心

须接收到整个分组。为了更为详细地探讨存储转发传输，考虑由两个端系统经一台路由器连接构成的简单网络，如图1-11所示。一台路由器通常有多条繁忙的链路，因为它的任务就是把一个入分组交换到一条出链路。在这个简单例子中，该路由器的任务相当简单：将分组从一条（输入）链路转移到另一条唯一的连接链路。在图1-11所示的特定时刻，源已经传输了分组1的一部分，分组1的前沿已经到达了路由器。因为该路由器应用了存储转发机制，所以此时它还不能传输已经接收的比特，而是必须先缓存（即“存储”）该分组的比特。仅当路由器已经接收完了该分组的所有比特后，它才能开始向出链路传输（即“转发”）该分组。为了深刻领悟存储转发传输，我们现在计算一下从源开始发送分组到目的地收到整个分组所经过的时间。（这里我们将忽略传播时延——指这些比特以接近光速的

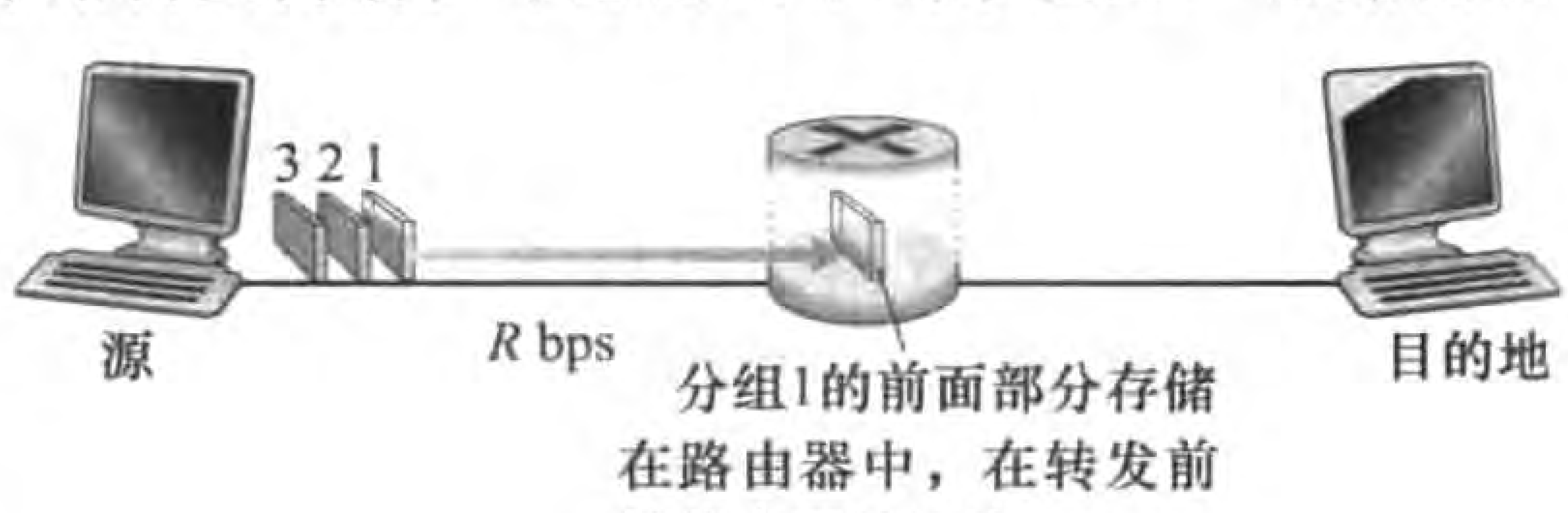


图1-11　存储转发分组交换

速度跨越线路所需要的时间，这将在 1.4 节讨论。）源在时刻 0 开始传输，在时刻 L/R 秒，因为该路由器刚好接收到整个分组，所以它能够朝着目的地向出链路开始传输分组；在时刻 $2L/R$，路由器已经传输了整个分组，并且整个分组已经被目的地接收。所以，总时延是 $2L/R$。如果交换机一旦比特到达就转发比特（不必首先收到整个分组），则因为比特没有在路由器保存，总时延将是 L/R。但是如我们将在 1.4 节中讨论的那样，路由器在转发前需要接收、存储和处理整个分组。

现在我们来计算从源开始发送第一个分组到目的地接收到所有三个分组所需的时间。与前面一样，在时刻 L/R，路由器开始转发第一个分组。而在时刻 L/R 源也开始发送第二个分组，因为它已经完成了第一个分组的完整发送。因此，在时刻 $2L/R$，目的地已经收到第一个分组并且路由器已经收到第二个分组。类似地，在时刻 $3L/R$，目的地已经收到前两个分组并且路由器已经收到第三个分组。最后，在时刻 $4L/R$，目的地已经收到所有 3 个分组！

我们现在来考虑下列一般情况：通过由 N 条速率均为 R 的链路组成的路径（所以，在源和目的地之间有 $N-1$ 台路由器），从源到目的地发送一个分组。应用如上相同的逻辑，我们看到端到端时延是：

$$d_{\text{端到端}} = N\frac{L}{R} \tag{1-1}$$

你也许现在要试着确定 P 个分组经过 N 条链路序列的时延有多大。

2. 排队时延和分组丢失

每台分组交换机有多条链路与之相连。对于每条相连的链路，该分组交换机具有一个**输出缓存**（output buffer，也称为**输出队列**（output queue）），它用于存储路由器准备发往那条链路的分组。该输出缓存在分组交换中起着重要的作用。如果到达的分组需要传输到某条链路，但发现该链路正忙于传输其他分组，该到达分组必须在输出缓存中等待。因此，除了存储转发时延以外，分组还要承受输出缓存的**排队时延**（queuing delay）。这些时延是变化的，变化的程度取决于网络的拥塞程度。因为缓存空间的大小是有限的，一个到达的分组可能发现该缓存已被其他等待传输的分组完全充满了。在此情况下，将出现**分组丢失（丢包）**（packet loss），到达的分组或已经排队的分组之一将被丢弃。

图 1-12 显示了一个简单的分组交换网络。如在图 1-11 中，分组被表示为三维厚片。厚片的宽度表示了该分组中比特的数量。在这张图中，所有分组具有相同的宽度，因此有相同的长度。假定主机 A 和 B 向主机 E 发送分组。主机 A 和 B 先通过 100Mbps 的以太网链路向第一个路由器发送分组。该路由器则将这些分组导向到一条 15Mbps 的链路。在某个短时间间隔内，如果分组到达路由器的到达率（转换为每秒比特）超过了 15Mbps，这些分组在通过链路传输之前，将在链路输出缓存中排队，在该路由器中将出现拥塞。例如，如果主机 A 和主机 B 每个都同时发送了 5 个紧接着的分组突发块，则这些分组中的大多数将在队列中等待一些时间。事实上，这完全类似于每天都在经历的一些情况，例如当我们在银行柜台前排队等待或在过路收费站前等待时。我们将在 1.4 节中更为详细地研究这种排队时延。

3. 转发表和路由选择协议

前面我们说过，路由器从与它相连的一条通信链路得到分组，然后向与它相连的另一条通信链路转发该分组。但是路由器怎样决定它应当向哪条链路进行转发呢？不同的计算

机网络实际上是以不同的方式完成分组转发的。这里，我们简要介绍在因特网中所采用的方法。

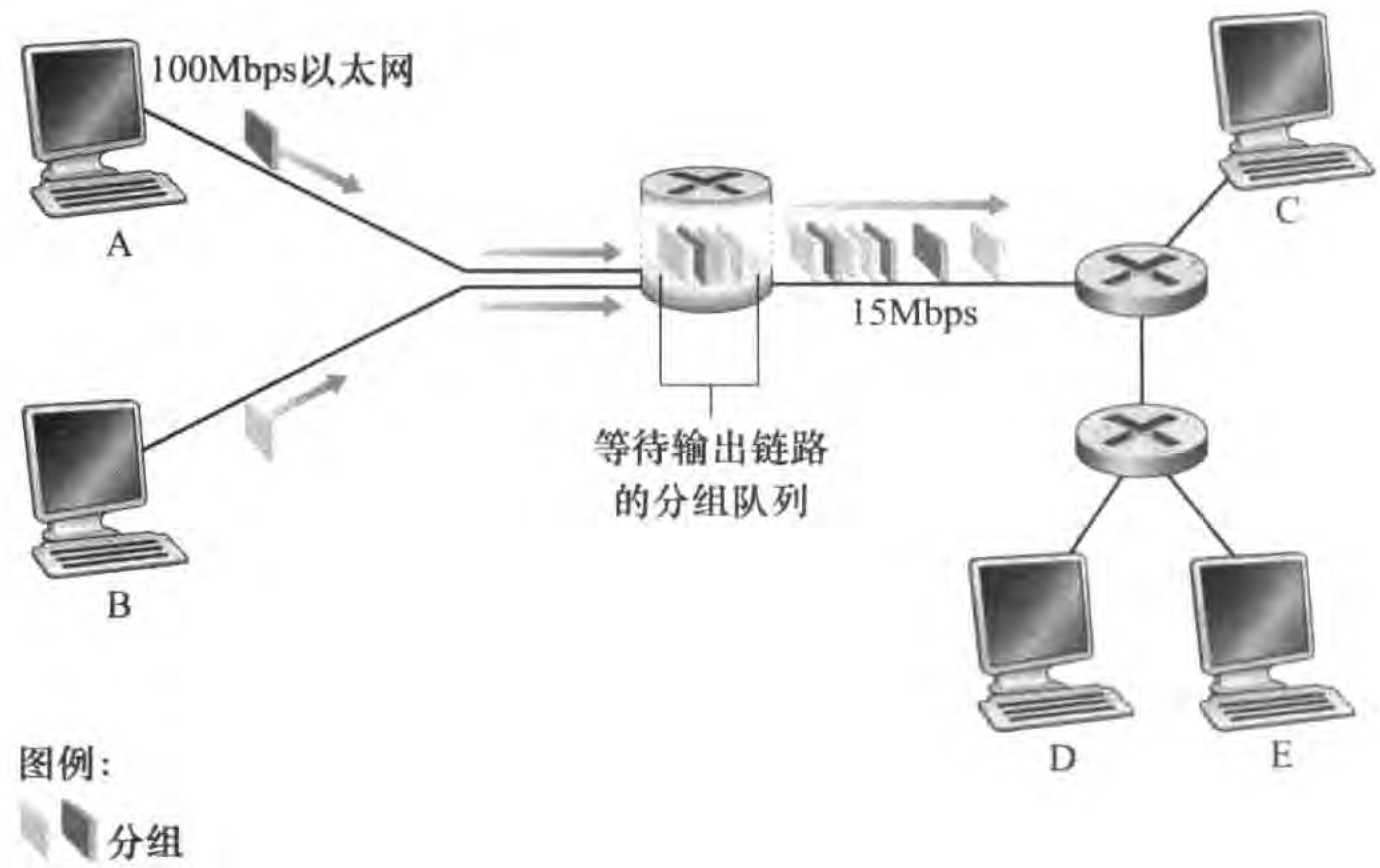


图1-12 分组交换

在因特网中，每个端系统具有一个称为IP地址的地址。当源主机要向目的端系统发送一个分组时，源在该分组的首部包含了目的地的IP地址。如同邮政地址那样，该地址具有一种等级结构。当一个分组到达网络中的路由器时，路由器检查该分组的目的地址的一部分，并向一台相邻路由器转发该分组。更特别的是，每台路由器具有一个**转发表**（forwarding table），用于将目的地址（或目的地址的一部分）映射成为输出链路。当某分组到达一台路由器时，路由器检查该地址，并用这个目的地址搜索其转发表，以发现适当的出链路。路由器则将分组导向该出链路。

端到端选路过程可以用一个不使用地图而喜欢问路的汽车驾驶员来类比。例如，假定Joe驾车从费城到佛罗里达州奥兰多市的Lakeside Drive街156号。Joe先驾车到附近的加油站，询问怎样才能到达佛罗里达州奥兰多市的Lakeside Drive街156号。加油站的服务员从该地址中抽取了佛罗里达州部分，告诉Joe他需要上I-95南州际公路，该公路恰有一个邻近该加油站的入口。他又告诉Joe，一旦到了佛罗里达后应当再问当地人。于是，Joe上了I-95南州际公路，一直到达佛罗里达的Jacksonville，在那里他向另一个加油站服务员问路。该服务员从地址中抽取了奥兰多市部分，告诉Joe他应当继续沿I-95公路到Daytona海滩，然后再问其他人。在Daytona海滩，另一个加油站服务员也抽取该地址的奥兰多部分，告诉Joe应当走I-4公路直接前往奥兰多。Joe走了I-4公路，并从奥兰多出口下来。Joe又向另一个加油站的服务员询问，这时该服务员抽取了该地址的Lakeside Drive部分，告诉了Joe到Lakeside Drive必须要走的路。一旦Joe到达了Lakeside Drive，他向一个骑自行车的小孩询问了到达目的地的方法。这个孩子抽取了该地址的156号部分，并指示了房屋的方向。Joe最后到达了最终目的地。在上述类比中，那些加油站服务员和骑车的孩子可类比为路由器。

我们刚刚学习了路由器使用分组的目的地址来索引转发表并决定适当的出链路。但是这个叙述还要求回答另一个问题：转发表是如何进行设置的？是通过人工对每台路由器逐台进行配置，还是因特网使用更为自动的过程进行配置呢？第5章将深入探讨这个问题。但在这里为了激发你的求知欲，我们现在将告诉你因特网具有一些特殊的**路由选择协议**

(routing protocol)，用于自动地设置这些转发表。例如，一个路由选择协议可以决定从每台路由器到每个目的地的最短路径，并使用这些最短路径结果来配置路由器中的转发表。

怎样才能实际看到分组在因特网中所走的端到端路由呢？我们现在请你亲手用一下Traceroute 程序。直接访问站点 www. traceroute. org，在一个特定的国家中选择一个源，跟踪从该源到你的计算机的路由。（参见 1.4 节有关 Traceroute 的讨论。）

1.3.2 电路交换

通过网络链路和交换机移动数据有两种基本方法：**电路交换**（circuit switching）和**分组交换**（packet switching）。上一小节已经讨论过分组交换网络，现在我们将注意力转向电路交换网络。

在电路交换网络中，在端系统间通信会话期间，预留了端系统间沿路径通信所需要的资源（缓存，链路传输速率）。在分组交换网络中，这些资源则不是预留的；会话的报文按需使用这些资源，其后果可能是不得不等待（即排队）接入通信线路。一个简单的类比是，考虑两家餐馆，一家需要顾客预订，而另一家不需要预订，但不保证能安排顾客。对于需要预订的那家餐馆，我们在离开家之前必须承受先打电话预订的麻烦，但当我们到达该餐馆时，原则上我们能够立即入座并点菜。对于不需要预订的那家餐馆，我们不必麻烦地预订餐桌，但当我们到达该餐馆时，也许不得不先等待一张餐桌空闲后才能入座。

传统的电话网络是电路交换网络的例子。考虑当一个人通过电话网向另一个人发送信息（语音或传真）时所发生的情况。在发送方能够发送信息之前，该网络必须在发送方和接收方之间建立一条连接。这是一个名副其实的连接，因为此时沿着发送方和接收方之间路径上的交换机都将为该连接维护连接状态。用电话的术语来说，该连接被称为一条**电路**(circuit)。当网络创建这种电路时，它也在连接期间在该网络链路上预留了恒定的传输速率（表示为每条链路传输容量的一部分）。既然已经为该发送方 - 接收方连接预留了带宽，则发送方能够以确保的恒定速率向接收方传送数据。

图 1-13 显示了一个电路交换网络。在这个网络中，用 4 条链路互联了 4 台电路交换机。这些链路中的每条都有 4 条电路，因此每条链路能够支持 4 条并行的连接。每台主机（例如 PC 和工作站）都与一台交换机直接相连。当两台主机要通信时，该网络在两台主机之间创建一条专用的**端到端连接**（end-to-end connection）。因此，主机 A 为了向主机 B 发送报文，网络必须在两条链路的每条上先预留一条电路。在这个例子中，这条专用的端到端连接使用第一条链路中的第二条电路和第二条链路中的第四条电路。因为每条链路具有 4 条电路，对于由端到端连接所使用的每条链路而言，该连接在连接期间获得链路总传输容量的 1/4。例如，如果两台邻近交换机之间每条链路具有 1Mbps 传输速率，则每个端到端电路交换连接获得 250kbps 专用的传输速率。

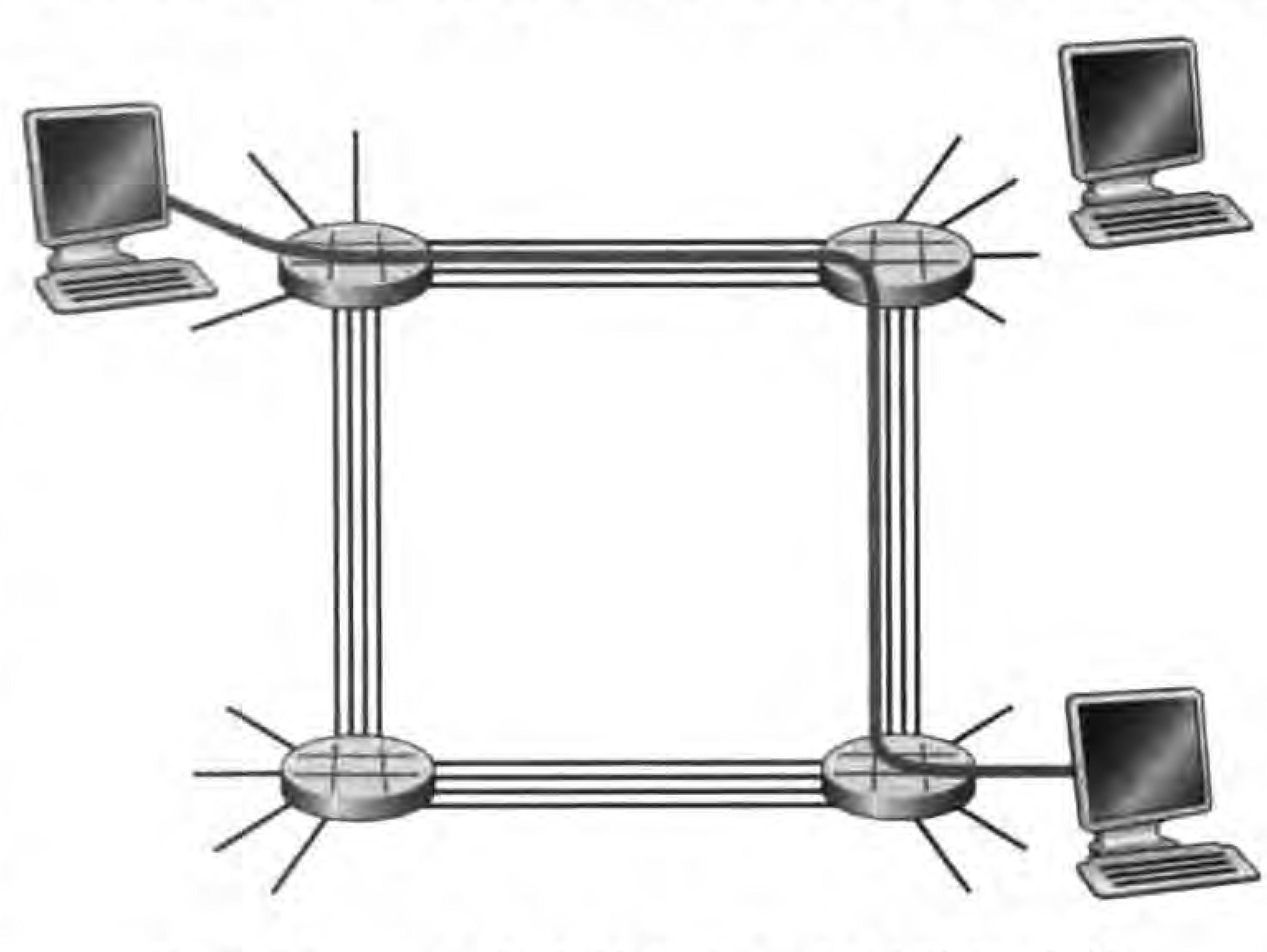


图 1-13　由 4 台交换机和 4 条链路组成的一个简单电路交换网络

与此相反，考虑一台主机要经过分组交换网络（如因特网）向另一台主机发送分组所发生的情况。与使用电路交换相同，该分组经过一系列通信链路传输。但与电路交换不同

的是，该分组被发送进网络，而不预留任何链路资源之类的东西。如果因为此时其他分组也需要经该链路进行传输而使链路之一出现拥塞，则该分组将不得不在传输链路发送侧的缓存中等待而产生时延。因特网尽最大努力以实时方式交付分组，但它不做任何保证。

1. 电路交换网络中的复用

链路中的电路是通过**频分复用**（Frequency-Division Multiplexing，FDM）或**时分复用**（Time-Division Multiplexing，TDM）来实现的。对于FDM，链路的频谱由跨越链路创建的所有连接共享。特别是，在连接期间链路为每条连接专用一个频段。在电话网络中，这个频段的宽度通常为4kHz（即每秒4000周期）。毫无疑问，该频段的宽度称为**带宽**（bandwidth）。调频无线电台也使用FDM来共享88MHz～108MHz的频谱，其中每个电台被分配一个特定的频段。

对于一条TDM链路，时间被划分为固定期间的帧，并且每个帧又被划分为固定数量的时隙。当网络跨越一条链路创建一条连接时，网络在每个帧中为该连接指定一个时隙。这些时隙专门由该连接单独使用，一个时隙（在每个帧内）可用于传输该连接的数据。

图1-14显示了一个支持多达4条电路的特定网络链路的FDM和TDM。对于FDM，其频率域被分割为4个频段，每个频段的带宽是4kHz。对于TDM，其时域被分割为帧，在每个帧中具有4个时隙，在循环的TDM帧中每条电路被分配相同的专用时隙。对于TDM，一条电路的传输速率等于帧速率乘以一个时隙中的比特数量。例如，如果链路每秒传输8000个帧，每个时隙由8个比特组成，则每条电路的传输速率是64kbps。

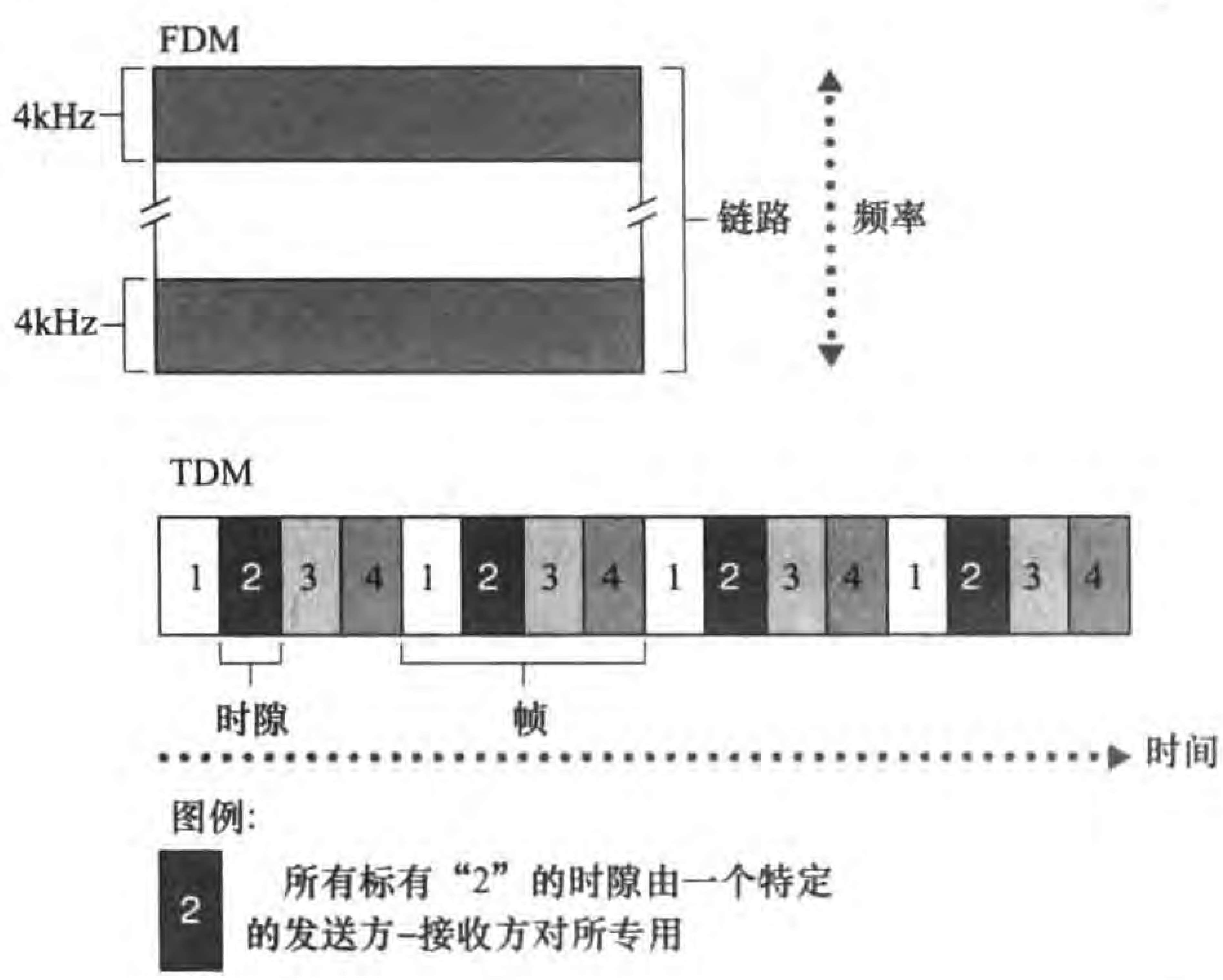


图1-14 对于FDM，每条电路连续地得到部分带宽。对于TDM，每条电路在短时间间隔（即时隙）中周期性地得到所有带宽

分组交换的支持者总是争辩说，电路交换因为在**静默期**（silent period）专用电路空闲而不够经济。例如，打电话的一个人停止讲话，空闲的网络资源（在沿该连接路由的链路中的频段或时隙）不能被其他进行中的连接所使用。作为这些资源不能有效利用的另一个例子，考虑一名放射科医师使用电路交换网络远程存取一系列X射线图像。该放射科医师建立一条连接，请求一幅图像，然后判读该图像，然后再请求一幅新图像。在放射科医师判读图像期间，网络资源分配给了该连接但没有使用（即被浪费了）。分组交换的支持者

还津津乐道地指出，创建端到端电路和预留端到端带宽是复杂的，需要复杂的信令软件以协调沿端到端路径的交换机的操作。

在结束讨论电路交换之前，我们讨论一个用数字表示的例子，它更能说明问题的实质。考虑从主机 A 到主机 B 经一个电路交换网络发送一个 640 000 比特的文件需要多长时间。假如在该网络中所有链路使用具有 24 时隙的 TDM，比特速率为 1.536Mbps。同时假定在主机 A 能够开始传输该文件之前，需要 500ms 创建一条端到端电路。它需要多长时间才能发送该文件？每条链路具有的传输速率是 1.536Mbps/24 = 64kbps，因此传输该文件需要（640kb）/(64kbps) = 10s。这个 10s，再加上电路创建时间，这样就需要 10.5s 发送该文件。值得注意的是，该传输时间与链路数量无关：端到端电路不管是通过一条链路还是 100 条链路，传输时间都将是 10s。（实际的端到端时延还包括传播时延，参见 1.4 节。）

2. 分组交换与电路交换的对比

在描述了电路交换和分组交换之后，我们来对比一下这两者。分组交换的批评者经常争辩说，分组交换不适合实时服务（例如，电话和视频会议），因为它的端到端时延是可变的和不可预测的（主要是因为排队时延的变动和不可预测所致）。分组交换的支持者却争辩道：①它提供了比电路交换更好的带宽共享；②它比电路交换更简单、更有效，实现成本更低。分组交换与电路交换之争的有趣讨论参见 [Molinero-Fernandez 2002]。概括而言，嫌餐馆预订麻烦的人宁可要分组交换而不愿意要电路交换。

分组交换为什么更有效呢？我们看一个简单的例子。假定多个用户共享一条 1Mbps 链路，再假定每个用户活跃周期是变化的，某用户时而以 100kbps 恒定速率产生数据，时而静止——这时用户不产生数据。进一步假定该用户仅有 10% 的时间活跃（余下的 90% 的时间空闲下来喝咖啡）。对于电路交换，在所有的时间内必须为每个用户预留 100kbps。例如，对于电路交换的 TDM，如果一个 1s 的帧被划分为 10 个时隙，每个时隙为 100ms，则每帧将为每个用户分配一个时隙。

因此，该电路交换链路仅能支持 10（= 1Mbps/100kbps）个并发的用户。对于分组交换，一个特定用户活跃的概率是 0.1（即 10%）。如果有 35 个用户，有 11 或更多个并发活跃用户的概率大约是 0.0004。（课后习题 P8 概述如何得到这个概率值。）当有 10 个或更少并发用户（以概率 0.9996 发生）时，到达的聚合数据速率小于或等于该链路的输出速率 1Mbps。因此，当有 10 个或更少的活跃用户时，通过该链路的分组流基本上没有时延，这与电路交换的情况一样。当同时活跃用户超过 10 个时，分组的聚合到达速率超过该链路的输出容量，则输出队列将开始变长。（一直增长到聚合输入速率重新低于 1Mbps，此后该队列长度才会减少。）因为在本例子中同时活跃用户超过 10 个的概率极小，分组交换差不多总是提供了与电路交换相同的性能，并且允许在用户数量是其 3 倍时情况也是如此。

我们现在考虑第二个简单的例子。假定有 10 个用户，某个用户突然产生 1000 个 1000 比特的分组，而其他用户则保持静默，不产生分组。在每帧具有 10 个时隙并且每个时隙包含 1000 比特的 TDM 电路交换情况下，活跃用户仅能使用每帧中的一个时隙来传输数据，而每个帧中剩余的 9 个时隙保持空闲。该活跃用户传输完所有 10^6 比特数据需要 10s 的时间。在分组交换情况下，活跃用户能够连续地以 1Mbps 的全部链路速率发送其分组，因为没有其他用户产生分组与该活跃用户的分组进行复用。在此情况下，该活跃用户的所

有数据将在1s内发送完毕。

上面的例子从两个方面表明了分组交换的性能能够优于电路交换的性能。这些例子也强调了在多个数据流之间共享链路传输速率的两种形式的关键差异。电路交换不考虑需求，而预先分配了传输链路的使用，这使得已分配而并不需要的链路时间未被利用。另一方面，分组交换按需分配链路使用。链路传输能力将在所有需要在链路上传输分组的用户之间逐分组地被共享。

虽然分组交换和电路交换在今天的电信网络中都是普遍采用的方式，但趋势无疑是朝着分组交换方向发展。甚至许多今天的电路交换电话网正在缓慢地向分组交换迁移。特别是，电话网经常在昂贵的海外电话部分使用分组交换。

1.3.3　网络的网络

我们在前面看到，端系统（PC、智能手机、Web服务器、电子邮件服务器等）经过一个接入ISP与因特网相连。该接入ISP能够提供有线或无线连接，使用了包括DSL、电缆、FTTH、WiFi和蜂窝等多种接入技术。值得注意的是，接入ISP不必是电信局或电缆公司，相反，它能够是如大学（为学生、教职员工和从业人员提供因特网接入）或公司（为其雇员提供接入）这样的单位。但让端用户和内容提供商连接到接入ISP仅解决了连接难题中的很小一部分，因为因特网是由数以亿计的用户构成的。要解决这个难题，接入ISP自身必须互联。通过创建*网络的网络*可以做到这一点，理解这个短语是理解因特网的关键。

年复一年，构成因特网的“网络的网络”已经演化成为一个非常复杂的结构。这种演化很大部分是由经济和国家策略驱动的，而不是由性能考虑驱动的。为了理解今天的因特网的网络结构，我们以逐步递进方式建造一系列网络结构，其中的每个新结构都更好地接近现在的复杂因特网。回顾前面互联接入ISP的中心目标，是使所有端系统能够彼此发送分组。一种幼稚的方法是使每个接入ISP*直接*与每个其他接入ISP连接。当然，这样的网状设计对于接入ISP费用太高，因为这将要求每个接入ISP要与世界上数十万个其他接入ISP有一条单独的通信链路。

我们的第一个网络结构即*网络结构1*，用单一的*全球传输ISP*互联所有接入ISP。我们假想的全球传输ISP是一个由路由器和通信链路构成的网络，该网络不仅跨越全球，而且至少具有一台路由器靠近数十万接入ISP中的每一个。当然，对于全球传输ISP，建造这样一个大规模的网络将耗资巨大。为了有利可图，自然要向每个连接的接入ISP收费，其价格反映（并不一定正比于）一个接入ISP经过全球ISP交换的流量大小。因为接入ISP向全球传输ISP付费，故接入ISP被认为是**客户**（customer），而全球传输ISP被认为是**提供商**（provider）。

如果某个公司建立并运营一个可赢利的全球传输ISP，其他公司建立自己的全球传输ISP并与最初的全球传输ISP竞争则是一件自然的事。这导致了*网络结构2*，它由数十万接入ISP和多个全球传输ISP组成。接入ISP无疑喜欢网络结构2胜过喜欢网络结构1，因为它们现在能够根据价格和服务因素在多个竞争的全球传输提供商之间进行选择。然而，值得注意的是，这些全球传输ISP之间必须是互联的；不然的话，与某个全球传输ISP连接的接入ISP将不能与连接到其他全球传输ISP的接入ISP进行通信。

刚才描述的网络结构2是一种两层的等级结构，其中全球传输提供商位于顶层，而接入ISP位于底层。这假设了全球传输ISP不仅能够接近每个接入ISP，而且发现经济上也

希望这样做。现实中，尽管某些 ISP 确实具有令人印象深刻的全球覆盖，并且确实直接与许多接入 ISP 连接，但世界上没有哪个 ISP 是无处不在的。相反，在任何给定的区域，可能有一个**区域 ISP**（regional ISP），区域中的接入 ISP 与之连接。每个区域 ISP 则与**第一层 ISP**（tier-1 ISP）连接。第一层 ISP 类似于我们假想的全球传输 ISP，尽管它不是在世界上每个城市中都存在，但它确实存在。有大约十几个第一层 ISP，包括 Level 3 Communications、AT&T、Sprint 和 NTT。有趣的是，没有组织正式认可第一层状态。俗话说：如果必须问你是否是一个组织的成员，你可能不是。

再来讨论这个网络的网络，不仅有多个竞争的第一层 ISP，而且在一个区域可能有多个竞争的区域 ISP。在这样的等级结构中，每个接入 ISP 向其连接的区域 ISP 支付费用，并且每个区域 ISP 向它连接的第一层 ISP 支付费用。（一个接入 ISP 也能直接与第一层 ISP 连接，这样它就向第一层 ISP 付费。）因此，在这个等级结构的每一层，都有客户 - 提供商关系。值得注意的是，第一层 ISP 不向任何人付费，因为它们位于该等级结构的顶部。更为复杂的情况是，在某些区域，可能有较大的区域 ISP（可能跨越整个国家），该区域中较小的区域 ISP 与之相连，较大的区域 ISP 则与第一层 ISP 连接。例如，在中国，每个城市有接入 ISP，它们与省级 ISP 连接，省级 ISP 又与国家级 ISP 连接，国家级 ISP 最终与第一层 ISP 连接 [Tian 2012]。这个多层等级结构仍然仅仅是今天因特网的粗略近似，我们称它为*网络结构 3*。

为了建造一个与今天的因特网更为相似的网络，我们必须在等级化网络结构 3 上增加存在点（Point of Presence，PoP）、多宿、对等和因特网交换点。PoP 存在于等级结构的所有层次，但底层（接入 ISP）等级除外。一个 **PoP** 只是提供商网络中的一台或多台路由器（在相同位置）群组，其中客户 ISP 能够与提供商 ISP 连接。对于要与提供商 PoP 连接的客户网络，它能从第三方电信提供商租用高速链路将它的路由器之一直接连接到位于该 PoP 的一台路由器。任何 ISP（除了第一层 ISP）可以选择**多宿**（multi-home），即可以与两个或更多提供商 ISP 连接。例如，一个接入 ISP 可能与两个区域 ISP 多宿，既可以与两个区域 ISP 多宿，也可以与一个第一层 ISP 多宿。当一个 ISP 多宿时，即使它的提供商之一出现故障，它仍然能够继续发送和接收分组。

正如我们刚才学习的，客户 ISP 向它们的提供商 ISP 付费以获得全球因特网互联能力。客户 ISP 支付给提供商 ISP 的费用数额反映了它通过提供商交换的通信流量。为了减少这些费用，位于相同等级结构层次的邻近一对 ISP 能够**对等**（peer），也就是说，能够直接将它们的网络连到一起，使它们之间的所有流量经直接连接而不是通过上游的中间 ISP 传输。当两个 ISP 对等时，通常不进行结算，即任一个 ISP 不向其对等付费。如前面提到的那样，第一层 ISP 也与另一个第一层 ISP 对等，它们之间无结算。对于对等和客户 - 提供商关系的讨论，[Van der Berg 2008] 是一本不错的读物。沿着这些相同路线，第三方公司能够创建一个**因特网交换点**（Internet Exchange Point，IXP），IXP 是一个汇合点，多个 ISP 能够在这里一起对等。IXP 通常位于一个有自己的交换机的独立建筑物中 [Ager 2012]，在今天的因特网中有 400 多个 IXP [IXP List 2016]。我们称这个生态系统为*网络结构 4*——由接入 ISP、区域 ISP、第一层 ISP、PoP、多宿、对等和 IXP 组成。

我们现在最终到达了*网络结构 5*，它描述了现今的因特网。在图 1-15 中显示了网络结构 5，它通过在网络结构 4 顶部增加**内容提供商网络**（content provider network）构建而成。谷歌是当前这样的内容提供商网络的一个突出例子。在本书写作之时，谷歌估计有50 ~ 100 个数据中心分布于北美、欧洲、亚洲、南美和澳大利亚。其中的某些数据中心容纳了

超过十万台的服务器，而另一些数据中心则较小，仅容纳数百台服务器。谷歌数据中心都经过专用的TCP/IP网络互联，该网络跨越全球，不过独立于公共因特网。重要的是，谷歌专用网络仅承载出入谷歌服务器的流量。如图1-15所示，谷歌专用网络通过与较低层ISP对等（无结算），尝试“绕过”因特网的较高层，采用的方式可以是直接与它们连接，或者在IXP处与它们连接［Labovitz 2010］。然而，因为许多接入ISP仍然仅能通过第一层网络的传输到达，所以谷歌网络也与第一层ISP连接，并就与这些ISP交换的流量向它们付费。通过创建自己的网络，内容提供商不仅减少了向顶层ISP支付的费用，而且对其服务最终如何交付给端用户有了更多的控制。谷歌的网络基础设施在2.6节中进行了详细描述。

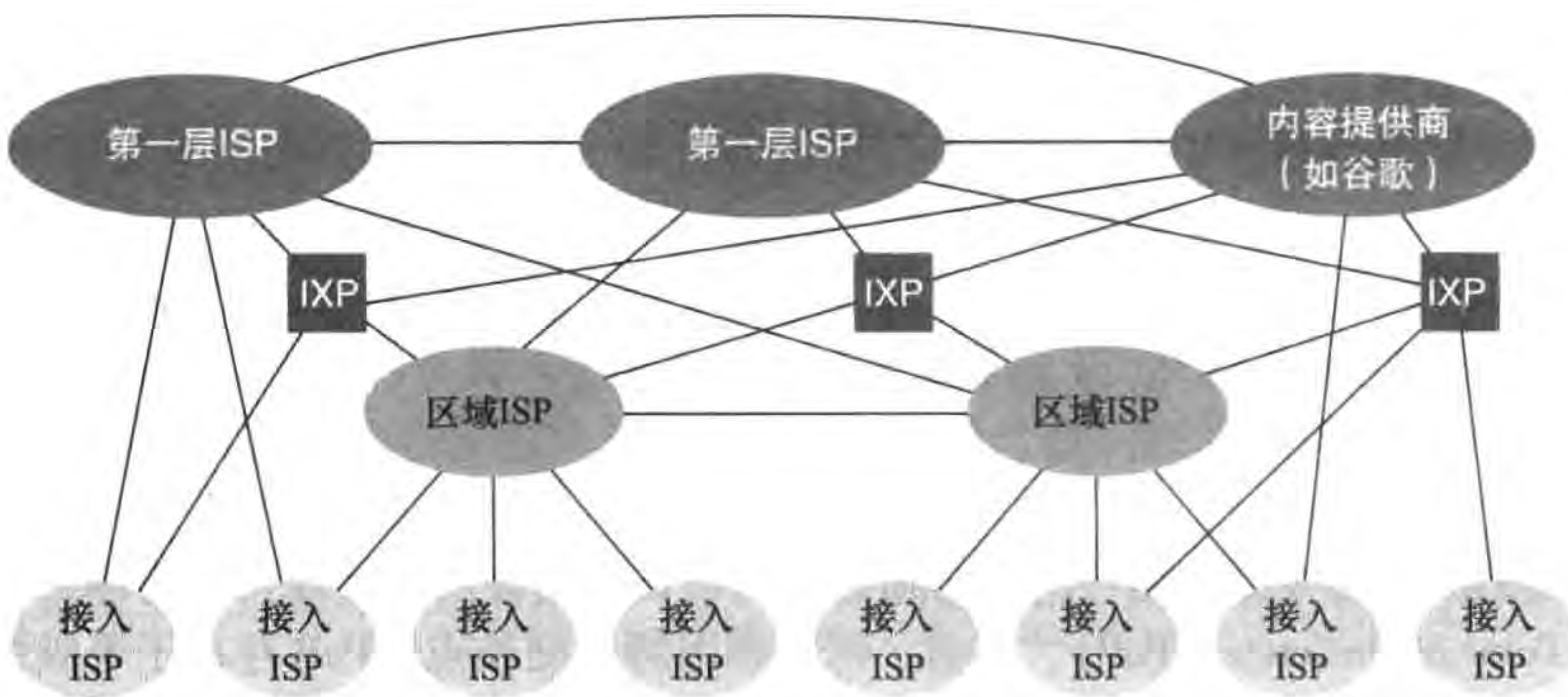


图1-15 ISP的互联

总结一下，今天的因特网是一个网络的网络，其结构复杂，由十多个第一层ISP和数十万个较低层ISP组成。ISP覆盖的区域多种多样，有些跨越多个大洲和大洋，有些限于狭窄的地理区域。较低层的ISP与较高层的ISP相连，较高层ISP彼此互联。用户和内容提供商是较低层ISP的客户，较低层ISP是较高层ISP的客户。近年来，主要的内容提供商也已经创建自己的网络，直接在可能的地方与较低层ISP互联。

1.4 分组交换网中的时延、丢包和吞吐量

回想在1.1节中我们讲过，因特网能够看成是一种基础设施，该基础设施为运行在端系统上的分布式应用提供服务。在理想情况下，我们希望因特网服务能够在任意两个端系统之间随心所欲地瞬间移动数据而没有任何数据丢失。然而，这是一个极高的目标，实践中难以达到。与之相反，计算机网络必定要限制在端系统之间的吞吐量（每秒能够传送的数据量），在端系统之间引入时延，而且实际上也会丢失分组。一方面，现实世界的物理定律引入的时延、丢包并限制吞吐量是不幸的。而另一方面，因为计算机网络存在这些问题，围绕如何去处理这些问题有许多令人着迷的话题，多得足以开设一门有关计算机网络方面的课程，可以做上千篇博士论文！在本节中，我们将开始研究和量化计算机网络中的时延、丢包和吞吐量等问题。

1.4.1 分组交换网中的时延概述

前面讲过，分组从一台主机（源）出发，通过一系列路由器传输，在另一台主机

(目的地)中结束它的历程。当分组从一个节点(主机或路由器)沿着这条路径到后继节点(主机或路由器),该分组在沿途的每个节点经受了几种不同类型的时延。这些时延最为重要的是**节点处理时延**(nodal processing delay)、**排队时延**(queuing delay)、**传输时延**(transmission delay)和**传播时延**(propagation delay),这些时延总体累加起来是**节点总时延**(total nodal delay)。许多因特网应用,如搜索、Web 浏览、电子邮件、地图、即时讯息和 IP 语音,它们的性能受网络时延的影响很大。为了深入理解分组交换和计算机网络,我们必须理解这些时延的性质和重要性。

时延的类型

我们来探讨一下图 1-16 环境中的这些时延。作为源和目的地之间的端到端路由的一部分,一个分组从上游节点通过路由器 A 向路由器 B 发送。我们的目标是在路由器 A 刻画出节点时延。值得注意的是,路由器 A 具有通往路由器 B 的出链路。该链路前面有一个队列(也称为缓存)。当分组从上游节点到达路由器 A 时,路由器 A 检查该分组的首部以决定它的适当出链路,并将该分组导向该链路。在这个例子中,对该分组的出链路是通向路由器 B 的那条链路。仅当在该链路没有其他分组正在传输并且没有其他分组排在该队列前面时,才能在这条链路上传输该分组;如果该链路当前正繁忙或有其他分组已经在该链路上排队,则新到达的分组将加入排队。

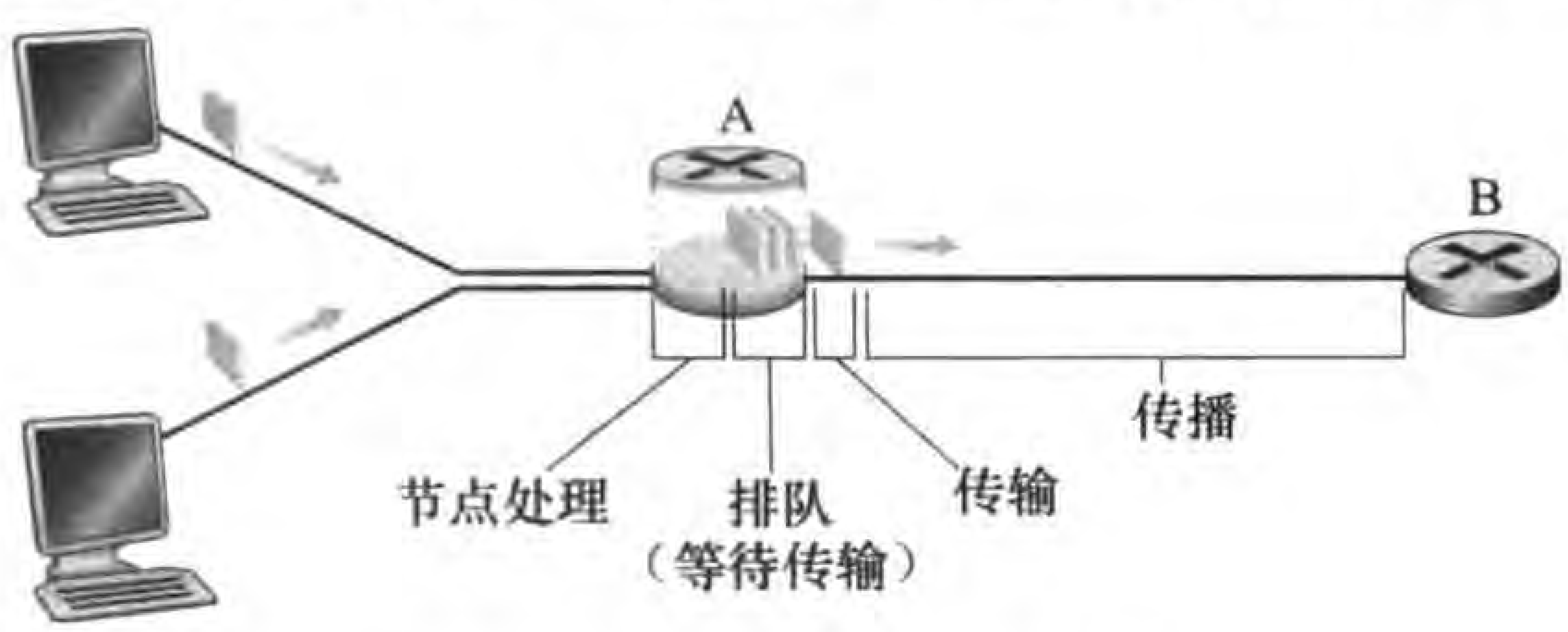


图 1-16 路由器 A 的节点时延

(1) 处理时延

检查分组首部和决定将该分组导向何处所需要的时间是**处理时延**的一部分。处理时延也能够包括其他因素,如检查比特级别的差错所需要的时间,该差错出现在从上游节点向路由器 A 传输这些分组比特的过程中。高速路由器的处理时延通常是微秒或更低的数量级。在这种节点处理之后,路由器将该分组引向通往路由器 B 链路之前的队列。(在第 4 章中,我们将研究路由器运行的细节。)

(2) 排队时延

在队列中,当分组在链路上等待传输时,它经受**排队时延**。一个特定分组的排队时延长度将取决于先期到达的正在排队等待向链路传输的分组数量。如果该队列是空的,并且当前没有其他分组正在传输,则该分组的排队时延为 0。另一方面,如果流量很大,并且许多其他分组也在等待传输,该排队时延将很长。我们将很快看到,到达分组期待发现的分组数量是到达该队列的流量的强度和性质的函数。实际的排队时延可以是毫秒到微秒量级。

(3) 传输时延

假定分组以先到先服务方式传输——这在分组交换网中是常见的方式,仅当所有已经到达的分组被传输后,才能传输刚到达的分组。用 L 比特表示该分组的长度,用 R bps(即 b/s)表示从路由器 A 到路由器 B 的链路传输速率。例如,对于一条 10Mbps 的以太网链路,速率 $R=10\text{Mbps}$;对于 100Mbps 的以太网链路,速率 $R=100\text{Mbps}$。**传输时延**是 L/R。这是将所有分组的比特推向链路(即传输,或者说发射)所需要的时间。实际的传输时延通常在毫秒到微秒量级。

（4）传播时延

一旦一个比特被推向链路，该比特需要向路由器B传播。从该链路的起点到路由器B传播所需要的时间是**传播时延**。该比特以该链路的传播速率传播。该传播速率取决于该链路的物理媒体（即光纤、双绞铜线等），其速率范围是$2\times10^8\sim3\times10^8$m/s，这等于或略小于光速。该传播时延等于两台路由器之间的距离除以传播速率。即传播时延是d/s，其中d是路由器A和路由器B之间的距离，s是该链路的传播速率。一旦该分组的最后一个比特传播到节点B，该比特及前面的所有比特被存储于路由器B。整个过程将随着路由器B执行转发而持续下去。在广域网中，传播时延为毫秒量级。

（5）传输时延和传播时延的比较

计算机网络领域的新手有时难以理解传输时延和传播时延之间的差异。该差异是微妙而重要的。传输时延是路由器推出分组所需要的时间，它是分组长度和链路传输速率的函数，而与两台路由器之间的距离无关。另一方面，传播时延是一个比特从一台路由器传播到另一台路由器所需要的时间，它是两台路由器之间距离的函数，而与分组长度或链路传输速率无关。

一个类比可以阐明传输时延和传播时延的概念。考虑一条公路每100km有一个收费站，如图1-17所示。可认为收费站间的公路段是链路，收费站是路由器。假定汽车以100km/h的速度（也就是说当一辆汽车离开一个收费站时，它立即加速到100km/h并在收费站间维持该速度）在该公路上行驶（即传播）。假定这时有10辆汽车作为一个车队在行驶，并且这10辆汽车以固定的顺序互相跟随。可以认为每辆汽车是一个比特，该车队是一个分组。同时假定每个收费站以每辆车12s的速度服务（即传输）一辆汽车，并且由于时间是深夜，因此该车队是公路上唯一一批汽车。最后，假定无论该车队的第一辆汽车何时到达收费站，它在入口处等待，直到其他9辆汽车到达并整队依次前行。（因此，整个车队在它能够“转发”之前，必须存储在收费站。）收费站将整个车队推向公路所需要的时间是（10辆车）/（5辆车/min）=2min。该时间类比于一台路由器中的传输时延。一辆汽车从一个收费站出口行驶到下一个收费站所需要的时间是100km/(100km/h)=1h。这个时间类比于传播时延。因此，从该车队存储在收费站前到该车队存储在下一个收费站前的时间是“传输时延”与“传播时间”总和，在本例中为62min。



图1-17 车队的类比

我们更深入地探讨一下这个类比。如果收费站对车队的服务时间大于汽车在收费站之间行驶的时间，将会发生什么情况呢？例如，假定现在汽车是以1000km/h的速率行驶，收费站是以每分钟一辆汽车的速率为汽车服务。则汽车在两个收费站之间的行驶时延是6min，收费站为车队服务的时间是10min。在此情况下，在该车队中的最后几辆汽车离开第一个收费站之前，该车队中前面的几辆汽车将会达到第二个收费站。这种情况在分组交换网中也会发生，一个分组中的前几个比特到达了一台路由器，而该分组中许多余下的比特仍然在前面的路由器中等待传输。

如果说一图胜千言的话，则一个动画必定胜百万言。与本书配套的Web网站提供了

一个交互式 Java 小程序，它很好地展现及对比了传输时延和传播时延。我们极力推荐读者访问该 Java 小程序。[Smith 2009] 也提供了可读性很好的有关传播、排队和传输时延的讨论。

如果令 d_{proc}、d_{queue}、d_{trans} 和 d_{prop} 分别表示处理时延、排队时延、传输时延和传播时延，则节点的总时延由下式给定：

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

这些时延成分所起的作用可能会有很大的不同。例如，对于连接两台位于同一个大学校园的路由器的链路而言，d_{prop} 可能是微不足道的（例如，几微秒）；然而，对于由同步卫星链路互联的两台路由器来说，d_{prop} 是几百毫秒，能够成为 d_{nodal} 中的主要成分。类似地，d_{trans} 的影响可能是微不足道的，也可能是很大的。通常对于 10Mbps 和更高的传输速率（例如，对于 LAN）的信道而言，它的影响是微不足道的；然而，对于通过低速拨号调制解调器链路发送的长因特网分组而言，可能是数百毫秒。处理时延 d_{proc} 通常是微不足道的；然而，它对一台路由器的最大吞吐量有重要影响，最大吞吐量是一台路由器能够转发分组的最大速率。

1.4.2 排队时延和丢包

节点时延的最为复杂和有趣的成分是排队时延 d_{queue}。事实上，排队时延在计算机网络中的重要程度及人们对它感兴趣的程度，从发表的数以千计的论文和大量专著的情况可见一斑 [Bertsekas 1991; Daigle 1991; Kleinrock 1975, 1976; Ross 1995]。我们这里仅给出有关排队时延的总体的、直觉的讨论；求知欲强的读者可能要浏览某些书籍（或者最终写有关这方面的博士论文）。与其他 3 项时延（即 d_{proc}、d_{trans} 和 d_{prop}）不同的是，排队时延对不同的分组可能是不同的。例如，如果 10 个分组同时到达空队列，传输的第一个分组没有排队时延，而传输的最后一个分组将经受相对大的排队时延（这时它要等待其他 9 个分组被传输）。因此，当表征排队时延时，人们通常使用统计量来度量，如平均排队时延、排队时延的方差和排队时延超过某些特定值的概率。

什么时候排队时延大，什么时候又不大呢？该问题的答案很大程度取决于流量到达该队列的速率、链路的传输速率和到达流量的性质，即流量是周期性到达还是以突发形式到达。为了更深入地领会某些要点，令 a 表示分组到达队列的平均速率（a 的单位是分组/秒，即 pkt/s）。前面讲过 R 是传输速率，即从队列中推出比特的速率（以 bps 即 b/s 为单位）。为了简单起见，也假定所有分组都是由 L 比特组成的。则比特到达队列的平均速率是 La bps。最后，假定该队列非常大，因此它基本能容纳无限数量的比特。比率 La/R 被称为**流量强度**（traffic intensity），它在估计排队时延的范围方面经常起着重要的作用。如果 $La/R > 1$，则比特到达队列的平均速率超过从该队列传输出去的速率。在这种不幸的情况下，该队列趋向于无限增加，并且排队时延将趋向无穷大！因此，流量工程中的一条金科玉律是：*设计系统时流量强度不能大于 1*。

现在考虑 $La/R \leq 1$ 时的情况。这时，到达流量的性质影响排队时延。例如，如果分组周期性到达，即每 L/R 秒到达一个分组，则每个分组将到达一个空队列中，不会有排队时延。另一方面，如果分组以突发形式到达而不是周期性到达，则可能会有很大的平均排队时延。例如，假定每 $(L/R)N$ 秒同时到达 N 个分组。则传输的第一个分组没有排队时延；传输的第二个分组就有 L/R 秒的排队时延；更为一般地，第 n 个传输的分组具有 $(n-1)$

L/*R* 秒的排队时延。我们将该例子中的计算平均排队时延的问题留给读者作为练习。

以上描述周期性到达的两个例子有些学术味。通常，到达队列的过程是随机的，即到达并不遵循任何模式，分组之间的时间间隔是随机的。在这种更为真实的情况下，量 *La*/*R* 通常不足以全面地表征时延的统计量。不过，直观地理解排队时延的范围很有用。特别是，如果流量强度接近于0，则几乎没有分组到达并且到达间隔很大，那么到达的分组将不可能在队列中发现别的分组。因此，平均排队时延将接近0。另一方面，当流量强度接近1时，当到达速率超过传输能力（由于分组到达速率的波动）时将存在时间间隔，在这些时段中将形成队列。当到达速率小于传输能力时，队列的长度将缩短。无论如何，随着流量强度接近1，平均排队长度变得越来越长。平均排队时延与流量强度的定性关系如图1-18所示。

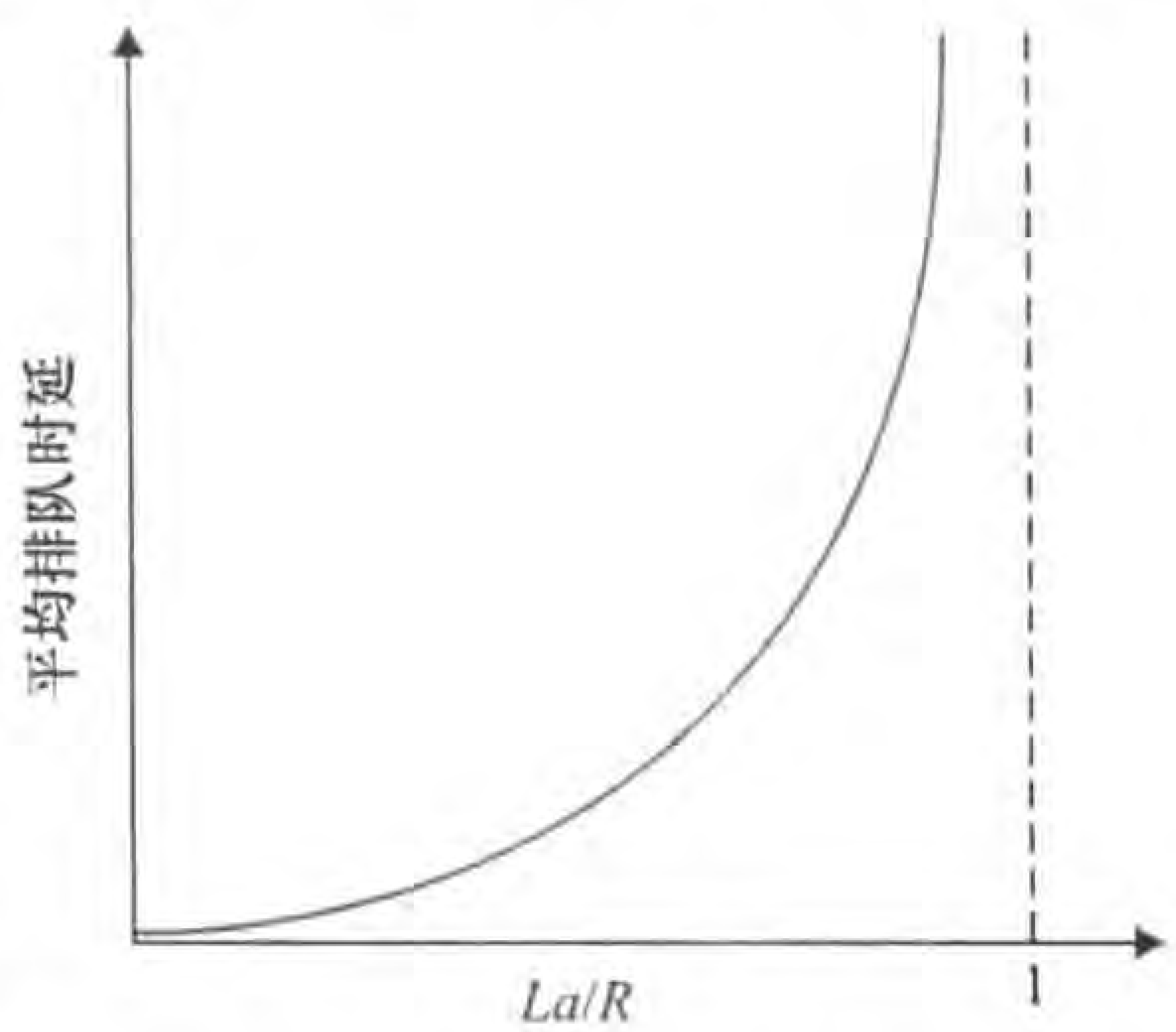


图1-18　平均排队时延与流量强度的关系

图1-18的一个重要方面是这样一个事实：随着流量强度接近于1，平均排队时延迅速增加。该强度的少量增加将导致时延大比例增加。也许你在公路上经历过这种事。如果在经常拥塞的公路上像平时一样驾驶，这条路经常拥塞的事实意味着它的流量强度接近于1，如果某些事件引起一个即便是稍微大于平常量的流量，经受的时延就可能很大。

为了实际感受一下排队时延的情况，我们再次鼓励你访问本书的Web网站，该网站提供了一个有关队列的交互式Java小程序。如果你将分组到达速率设置得足够大，使流量强度超过1，那么将看到经过一段时间后，队列慢慢地建立起来。

丢包

在上述讨论中，我们已经假设队列能够容纳无穷多的分组。在现实中，一条链路前的队列只有有限的容量，尽管排队容量极大地依赖于路由器设计和成本。因为该排队容量是有限的，随着流量强度接近1，排队时延并不真正趋向无穷大。相反，到达的分组将发现一个满的队列。由于没有地方存储这个分组，路由器将**丢弃**（drop）该分组，即该分组将会**丢失**（lost）。当流量强度大于1时，队列中的这种溢出也能够在用于队列的Java小程序中看到。

从端系统的角度看，上述丢包现象看起来是一个分组已经传输到网络核心，但它绝不会从网络发送到目的地。分组丢失的比例随着流量强度增加而增加。因此，一个节点的性能常常不仅根据时延来度量，而且根据丢包的概率来度量。正如我们将在后面各章中讨论的那样，丢失的分组可能基于端到端的原则重传，以确保所有的数据最终从源传送到了目的地。

1.4.3　端到端时延

前面的讨论一直集中在节点时延上，即在单台路由器上的时延。我们现在考虑从源到目的地的总时延。为了能够理解这个概念，假定在源主机和目的主机之间有 $N-1$ 台路由器。我们还要假设该网络此时是无拥塞的（因此排队时延是微不足道的），在每台路由器和源主机上的处理时延是 d_{proc}，每台路由器和源主机的输出速率是 *R* bps，每条链路的传

播时延是 d_{prop}。节点时延累加起来，得到端到端时延：

$$d_{end-end} = N(d_{proc} + d_{trans} + d_{prop}) \tag{1-2}$$

同样，式中 $d_{trans} = L/R$，其中 L 是分组长度。值得注意的是，式（1-2）是式（1-1）的一般形式，式（1-1）没有考虑处理时延和传播时延。在各节点具有不同的时延和每个节点存在平均排队时延的情况下，需要对式（1-2）进行一般化处理。我们将有关工作留给读者。

1. Traceroute

为了对计算机网络中的端到端时延有第一手认识，我们可以利用 Traceroute 程序。Traceroute 是一个简单的程序，它能够在任何因特网主机上运行。当用户指定一个目的主机名字时，源主机中的该程序朝着目的地发送多个特殊的分组。当这些分组向着目的地传送时，它们通过一系列路由器。当路由器接收到这些特殊分组之一时，它向源回送一个短报文。该短报文包括路由器的名字和地址。

更具体的是，假定在源和目的地之间有 $N-1$ 台路由器。源将向网络发送 N 个特殊的分组，其中每个分组地址指向最终目的地。这 N 个特殊分组标识为从 1 到 N，第一个分组标识为 1，最后的分组标识为 N。当第 n 台路由器接收到标识为 n 的第 n 个分组时，该路由器不是向它的目的地转发该分组，而是向源回送一个报文。当目的主机接收第 N 个分组时，它也会向源返回一个报文。该源记录了从它发送一个分组到它接收到对应返回报文所经历的时间；它也记录了返回该报文的路由器（或目的主机）的名字和地址。以这种方式，源能够重建分组从源到目的地所采用的路由，并且该源能够确定到所有中间路由器的往返时延。Traceroute 实际上对刚才描述的实验重复了 3 次，因此该源实际上向目的地发送了 $3 \times N$ 个分组。RFC 1393 详细地描述了 Traceroute。

这里有一个 Traceroute 程序输出的例子，其中追踪的路由从源主机 gaia. cs. umass. edu（位于马萨诸塞大学）到 cis. poly. edu（位于布鲁克林的理工大学）。输出有 6 列：第一列是前面描述的 n 值，即路径上的路由器编号；第二列是路由器的名字；第三列是路由器地址（格式为 xxx. xxx. xxx. xxx）；最后 3 列是 3 次实验的往返时延。如果源从任何给定路由器接收到的报文少于 3 条（由于网络中的丢包），Traceroute 在该路由器号码后面放一个星号，并向那台路由器报告少于 3 次往返时间。

```
1  cs-gw (128.119.240.254) 1.009 ms 0.899 ms 0.993 ms
2  128.119.3.154 (128.119.3.154) 0.931 ms 0.441 ms 0.651 ms
3  -border4-rt-gi-1-3.gw.umass.edu (128.119.2.194) 1.032 ms 0.484 ms 0.451 ms
4  -acr1-ge-2-1-0.Boston.cw.net (208.172.51.129) 10.006 ms 8.150 ms 8.460 ms
5  -agr4-loopback.NewYork.cw.net (206.24.194.104) 12.272 ms 14.344 ms 13.267 ms
6  -acr2-loopback.NewYork.cw.net (206.24.194.62) 13.225 ms 12.292 ms 12.148 ms
7  -pos10-2.core2.NewYork1.Level3.net (209.244.160.133) 12.218 ms 11.823 ms 11.793 ms
8  -gige9-1-52.hsipaccess1.NewYork1.Level3.net (64.159.17.39) 13.081 ms 11.556 ms 13.297 ms
9  -p0-0.polyu.bbnplanet.net (4.25.109.122) 12.716 ms 13.052 ms 12.786 ms
10 cis.poly.edu (128.238.32.126) 14.080 ms 13.035 ms 12.802 ms
```

在上述跟踪中，在源和目的之间有 9 台路由器。这些路由器中的多数有一个名字，所有都有地址。例如，路由器 3 的名字是 border4- rt- gi- 1- 3. gw. umass. edu，它的地址是 128. 119. 2. 194。看看为这台路由器提供的数据，可以看到在源和路由器之间的往返时延：3 次实验中的第一次是 1. 03ms，后继两次实验的往返时延是 0. 48ms 和 0. 45ms。这些往返时延包括刚才讨论的所有时延，即包括传输时延、传播时延、路由器处理时延和排队时延。因为该排队时延随时间变化，所以分组 n 发送到路由器 n 的往返时延实际上可能比分

组$n+1$发送到路由器$n+1$的往返时延更长。的确，我们在上述例子中观察到了这种现象：到路由器6的时延比到路由器7的更大！

你想自己试试Traceroute程序吗？我们极力推荐你访问 http://www.traceroute.org，它的Web界面提供了有关路由跟踪的大量源列表。你选择一个源，并提供任何目的地的主机名，该Traceroute程序则会完成所有工作。有许多为Traceroute提供图形化界面的免费软件，其中我们喜爱的一个程序是PingPlotter[PingPlotter 2016]。

2. 端系统、应用程序和其他时延

除了处理时延、传输时延和传播时延，端系统中还有其他一些重要时延。例如，希望向共享媒体（例如在WiFi或电缆调制解调器情况下）传输分组的端系统可能有意地延迟它的传输，把这作为它与其他端系统共享媒体的协议的一部分；我们将在第6章中详细地考虑这样的协议。另一个重要的时延是媒体分组化时延，这种时延出现在IP语音（VoIP）应用中。在VoIP中，发送方在向因特网传递分组之前必须首先用编码的数字化语音填充一个分组。这种填充一个分组的时间称为分组化时延，它可能较大，并能够影响用户感受到的VoIP呼叫的质量。这个问题将在本章结束的课后作业中进一步探讨。

1.4.4 计算机网络中的吞吐量

除了时延和丢包，计算机网络中另一个至关重要的性能测度是端到端吞吐量。为了定义吞吐量，考虑从主机A到主机B跨越计算机网络传送一个大文件。例如，也许是从一个P2P文件共享系统中的一个对等方向另一个对等方传送一个大视频片段。在任何时间瞬间的**瞬时吞吐量**（instantaneous throughput）是主机B接收到该文件的速率（以bps计）。（许多应用程序包括许多P2P文件共享系统，其用户界面显示了下载期间的瞬时吞吐量，也许你以前已经观察过它！）如果该文件由F比特组成，主机B接收到所有F比特用去T秒，则文件传送的**平均吞吐量**（average throughput）是F/T bps。对于某些应用程序如因特网电话，希望具有低时延和在某个阈值之上（例如，对某些因特网电话是超过24kbps，对某些实时视频应用程序是超过256kbps）的一致的瞬时吞吐量。对于其他应用程序，包括涉及文件传送的那些应用程序，时延不是决定性的，但是希望具有尽可能高的吞吐量。

为了进一步深入理解吞吐量这个重要概念，我们考虑几个例子。图1-19a显示了服务器和客户这两个端系统，它们由两条通信链路和一台路由器相连。考虑从服务器传送一个文件到客户的吞吐量。令R_s表示服务器与路由器之间的链路速率；R_c表示路由器与客户之间的链路速率。假定在整个网络中只有从该服务器到客户的比特在传送。在这种理想的情况下，我们要问该服务器到客户的吞吐量是多少？为了回答这个问题，我们可以想象比特是流体，通信链路是管道。显然，这台服务器不能以快于R_s bps的速率通过其链路注入比特；这台路由器也不能以快于R_c bps的速率转发比特。如果$R_s<R_c$，则在给定的吞吐量R_s bps的情况下，由该服务器注入的比特将顺畅地通过路由器“流动”，并以速率R_s bps到达客户。另一方面，如果$R_c<R_s$，则该路由器将不能像接收速率那样快地转发比特。在这种情况下，比特将以速率R_c离开该路由器，从而得到端到端吞吐量R_c。（还要注意的是，如果比特继续以速率R_s到达路由器，继续以R_c离开路由器的话，在该路由器中等待传输给客户的积压比特将不断增加，这是一种最不希望的情况！）因此，对于这种简单的两链路网络，其吞吐量是$\min\{R_c, R_s\}$，这就是说，它是**瓶颈链路**（bottleneck link）的传输速率。在决定了吞吐量之后，我们现在近似地得到从服务器到

客户传输一个 F 比特的大文件所需要的时间是 $F/\min\{R_c, R_s\}$。举一个特定的例子，假定你正在下载一个 $F=32\times10^6$ 比特的 MP3 文件，服务器具有 $R_s=2$Mbps 的传输速率，并且你有一条 $R_c=1$Mbps 的接入链路。则传输该文件所需的时间是 32 秒。当然，这些吞吐量和传输时间的表达式仅是近似的，因为它们并没有考虑存储转发、处理时延和协议等问题。

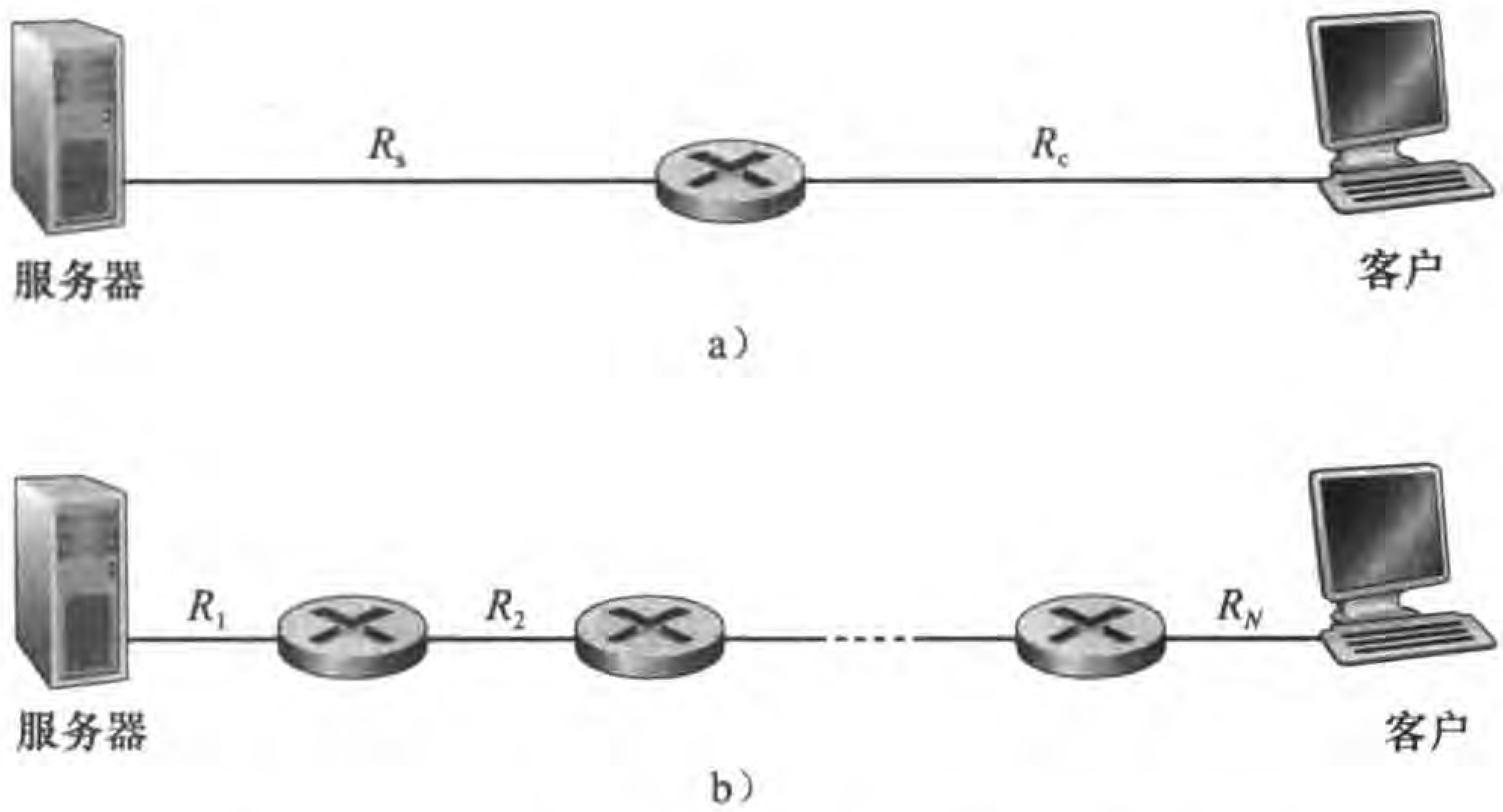


图 1-19　一个文件从服务器传送到客户的吞吐量

图 1-19b 此时显示了一个在服务器和客户之间具有 N 条链路的网络，这 N 条链路的传输速率分别是 R_1，R_2，…，R_N。应用对两条链路网络的分析方法，我们发现从服务器到客户的文件传输吞吐量是 $\min\{R_1, R_2, \cdots, R_N\}$，这同样仍是沿着服务器和客户之间路径的瓶颈链路的速率。

现在考虑由当前因特网所引发的另一个例子。图 1-20a 显示了与一个计算机网络相连的两个端系统：一台服务器和一个客户。考虑从服务器向客户传送一个文件的吞吐量。服务器以速率为 R_s 的接入链路与网络相连，且客户以速率为 R_c 的接入链路与网络相连。现在假定在通信网络核心中的所有链路具有非常高的传输速率，即该速率比 R_s 和 R_c 要高得多。目前因特网的核心的确超量配置了高速率的链路，从而很少出现拥塞。同时假定在整个网络中发送的比特都是从该服务器到该客户。在这个例子中，因为计算机网络的核心就像一个粗大的管子，所以比特从源向目的地的流动速率仍是 R_s 和 R_c 中的最小者，即吞吐量 $=\min\{R_s, R_c\}$。因此，在今天因特网中对吞吐量的限制因素通常是接入网。

作为最后一个例子，考虑图 1-20b，其中有 10 台服务器和 10 个客户与某计算机网络核心相连。在这个例子中，同时发生 10 个下载，涉及 10 个客户－服务器对。假定这 10 个下载是网络中当时的唯一流量。如该图所示，在核心中有一条所有 10 个下载通过的链路。将这条链路 R 的传输速率表示为 R。假定所有服务器接入链路具有相同的速率 R_s，所有客户接入链路具有相同的速率 R_c，并且核心中除了速率为 R 的一条共同链路之外的所有链路，它们的传输速率都比 R_s、R_c 和 R 大得多。现在我们要问，这种下载的吞吐量是多少？显然，如果该公共链路的速率 R 很大，比如说比 R_s 和 R_c 大 100 倍，则每个下载的吞吐量将仍然是 $\min\{R_s, R_c\}$。但是如果该公共链路的速率与 R_s 和 R_c 有相同量级会怎样呢？在这种情况下其吞吐量将是多少呢？让我们观察一个特定的例子。假定 $R_s=2$Mbps，$R_c=1$Mbps，$R=5$Mbps，并且公共链路为 10 个下载平等划分它的传输速率。这时每个下载的瓶颈不再位于接入网中，而是位于核心中的共享链路了，该瓶颈仅能为每个下载提供 500kbps 的吞吐量。因此每个下载的端到端吞吐量现在减少到 500kbps。

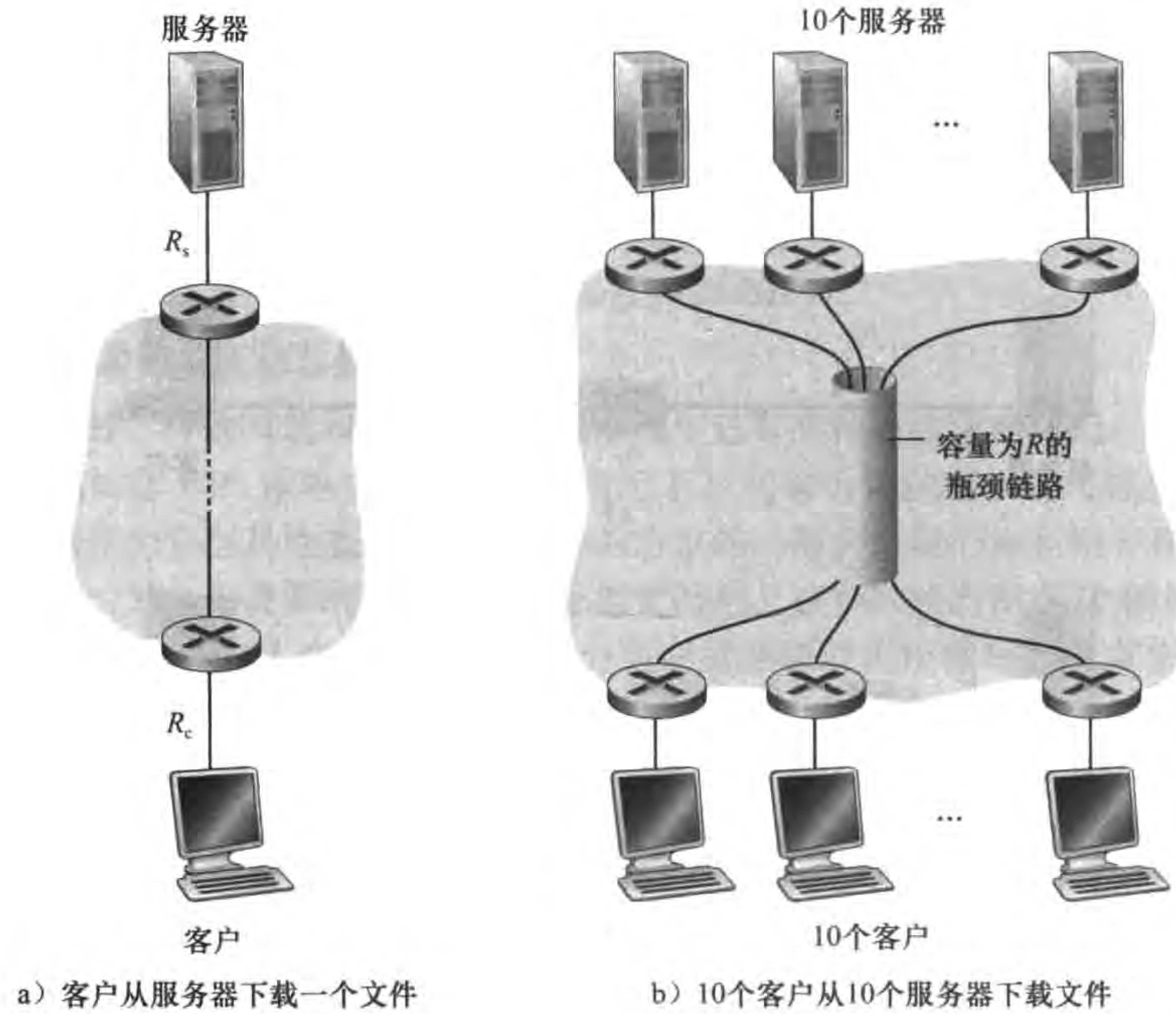


图 1-20 端到端吞吐量

图 1-19 和图 1-20 中的例子说明吞吐量取决于数据流过的链路的传输速率。我们看到当没有其他干扰流量时，其吞吐量能够近似为沿着源和目的地之间路径的最小传输速率。图 1-20b 中的例子更一般地说明了吞吐量不仅取决于沿着路径的传输速率，而且取决于干扰流量。特别是，如果许多其他的数据流也通过这条链路流动，一条具有高传输速率的链路仍然可能成为文件传输的瓶颈链路。我们将在课后习题中和后继章节中更仔细地研究计算机网络中的吞吐量。

1.5 协议层次及其服务模型

从我们到目前的讨论来看，因特网显然是一个极为复杂的系统。我们已经看到，因特网有许多部分：大量的应用程序和协议、各种类型的端系统、分组交换机以及各种类型的链路级媒体。面对这种巨大的复杂性，存在着组织网络体系结构的希望吗？或者至少存在着我们对网络体系结构进行讨论的希望吗？幸运的是，对这两个问题的回答都是肯定的。

1.5.1 分层的体系结构

在试图组织我们关于因特网体系结构的想法之前，先看一个人类社会与之类比的例子。实际上，在日常生活中我们一直都与复杂系统打交道。想象一下有人请你描述比如航线系统的情况吧。你怎样用一个结构来描述这样一个复杂的系统？该系统具有票务代理、行李检查、登机口人员、飞行员、飞机、空中航行控制和世界范围的导航系统。描述这种系统的一种方式是，描述当你乘某个航班时，你（或其他人替你）要采取的一系列动作。你要购买机票，托运行李，去登机口，并最终登上这次航班。该飞机起飞，飞行到目的

地。当飞机着陆后，你从登机口离机并认领行李。如果这次行程不理想，你向票务机构投诉这次航班（你的努力一无所获）。图 1-21 显示了相关情况。

我们已经能从这里看出与计算机网络的某些类似：航空公司把你从源送到目的地；而分组被从因特网中的源主机送到目的主机。但这不是我们寻求的完全的类似。我们在图 1-21 中寻找某些结构。观察图 1-21，我们注意到在每一端都有票务功能；还对已经检票的乘客有托运行李功能，对已经检票并已经检查过行李的乘客有登机功能。对于那些已经通过登机口的乘客（即已经经过检票、行李检查和通过登机口的乘客），有起飞和着陆的功能，并且在飞行中，有飞机按预定路线飞行的功能。这提示我们能够以水平的方式看待这些功能，如图 1-22 所示。



图 1-21　乘飞机旅行的一系列动作

图 1-22 将航线功能划分为一些层次，提供了我们能够讨论航线旅行的框架。注意到每个层次与其下面的层次结合在一起，实现了某些功能、服务。在票务层及以下，完成了一个人从航线柜台到航线柜台的转移。在行李层及以下，完成了人和行李从行李托运到行李认领的转移。注意到行李层仅对已经完成票务的人提供服务。在登机口层，完成了人和行李从离港登机口到到港登机口的转移。在起飞/着陆层，完成了一个人和手提行李从跑道到跑道的转移。每个层次通过以下方式提供服务：①在这层中执行了某些动作（例如，在登机口层，某飞机的乘客登机和离机）；②使用直接下层的服务（例如，在登机口层，使用起飞/着陆层的跑道到跑道的旅客转移服务）。

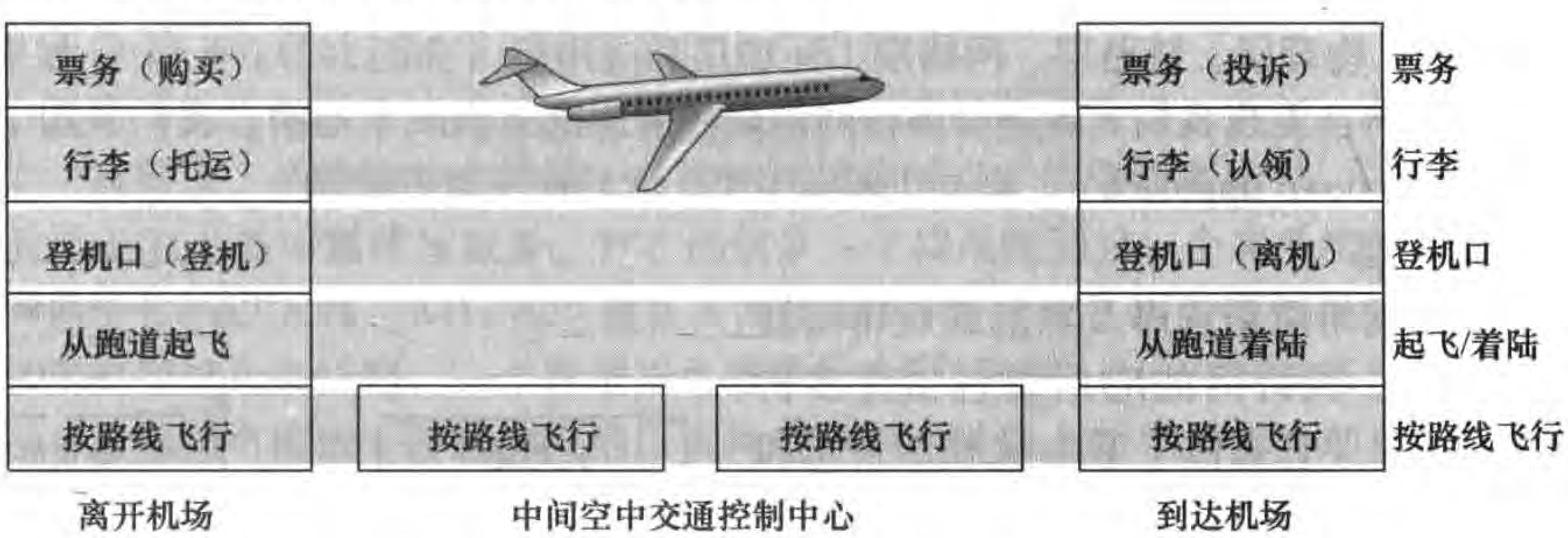


图 1-22　航线功能的水平分层

利用分层的体系结构，我们可以讨论一个大而复杂系统的定义良好的特定部分。这种简化本身由于提供模块化而具有很高价值，这使某层所提供的服务实现易于改变。只要该层对其上面的层提供相同的服务，并且使用来自下面层次的相同服务，当某层的实现变化时，该系统的其余部分保持不变。（注意到改变一个服务的实现与改变服务本身是极为不同的！）例如，如果登机口功能被改变了（例如让人们按身高登机和离机），航线系统的其余部分将保持不变，因为登机口仍然提供相同的功能（人们登机和离机）；改变后，它仅是以不同的方式实现了该功能。对于大而复杂且需要不断更新的系统，改变服务的实现而不影响该系统其他组件是分层的另一个重要优点。

1. 协议分层

我们对航线已经进行了充分讨论，现将注意力转向网络协议。为了给网络协议的设计提供一个结构，网络设计者以**分层**（layer）的方式组织协议以及实现这些协议的网络硬件和软件。每个协议属于这些层次之一，就像图1-22所示的航线体系结构中的每种功能属于某一层一样。我们再次关注某层向它的上一层提供的**服务**（service），即所谓一层的**服务模型**（service model）。就像前面航线例子中的情况一样，每层通过在该层中执行某些动作或使用直接下层的服务来提供服务。例如，由第 n 层提供的服务可能包括报文从网络的一边到另一边的可靠交付。这可能是通过使用第 $n-1$ 层的边缘到边缘的不可靠报文传送服务，加上第 n 层的检测和重传丢失报文的功能来实现的。

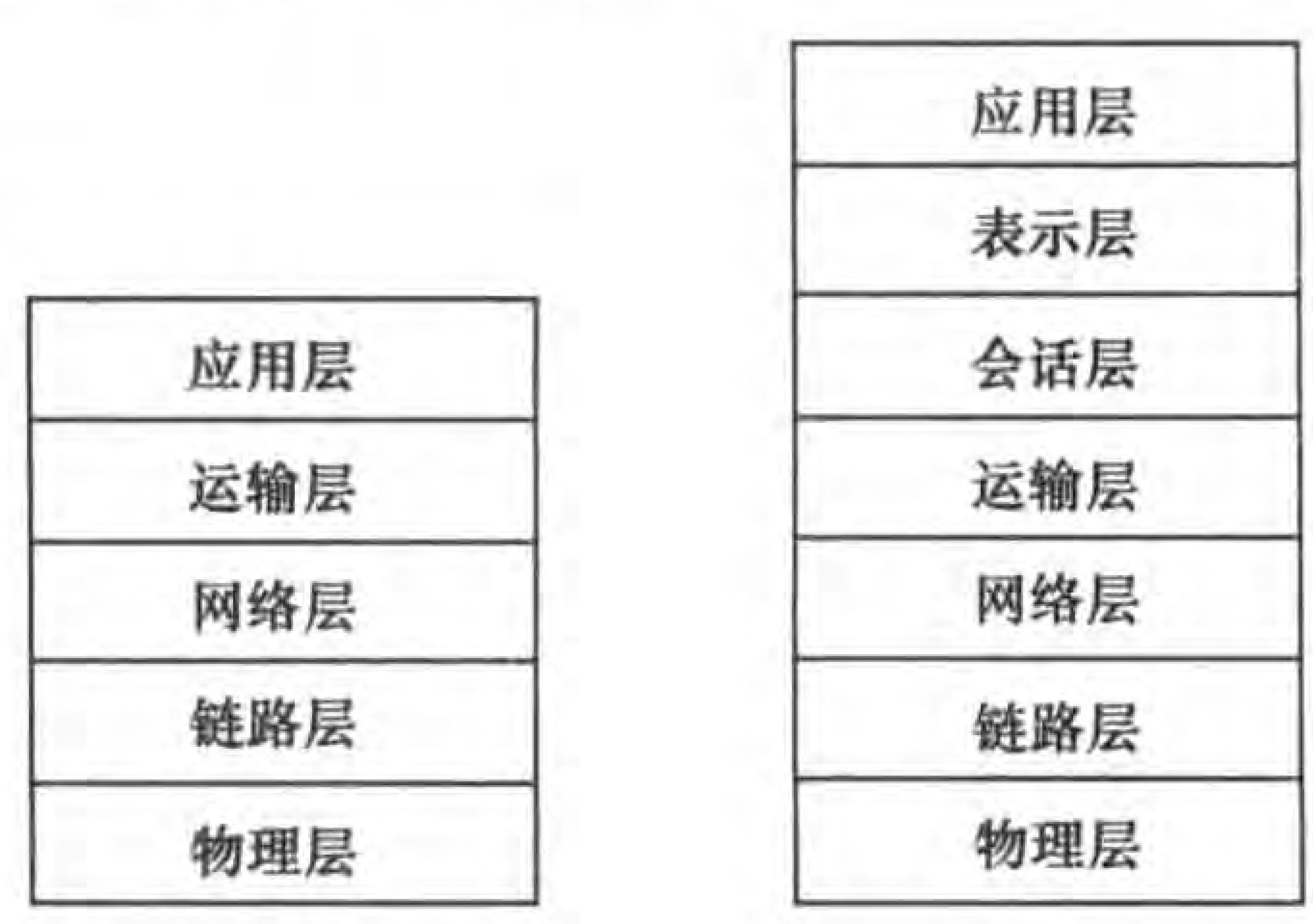
一个协议层能够用软件、硬件或两者的结合来实现。诸如HTTP和SMTP这样的应用层协议几乎总是在端系统中用软件实现，运输层协议也是如此。因为物理层和数据链路层负责处理跨越特定链路的通信，它们通常在与给定链路相关联的网络接口卡（例如以太网或WiFi接口卡）中实现。网络层经常是硬件和软件实现的混合体。还要注意的是，如同分层的航线体系结构中的功能分布在构成该系统的各机场和飞行控制中心中一样，一个第 n 层协议也分布在构成该网络的端系统、分组交换机和其他组件中。这就是说，第 n 层协议的不同部分常常位于这些网络组件的各部分中。

协议分层具有概念化和结构化的优点［RFC 3439］。如我们看到的那样，分层提供了一种结构化方式来讨论系统组件。模块化使更新系统组件更为容易。然而，需要提及的是，某些研究人员和联网工程师激烈地反对分层［Wakeman 1992］。分层的一个潜在缺点是一层可能冗余较低层的功能。例如，许多协议栈在基于每段链路和基于端到端两种情况下，都提供了差错恢复。第二种潜在的缺点是某层的功能可能需要仅在其他某层才出现的信息（如时间戳值），这违反了层次分离的目标。

将这些综合起来，各层的所有协议被称为**协议栈**（protocol stack）。因特网的协议栈由5个层次组成：物理层、链路层、网络层、运输层和应用层（如图1-23a所示）。如果你查看本书目录，将会发现我们大致是以因特网协议栈的层次来组织本书的。我们采用了**自顶向下方法**（top-down approach），首先处理应用层，然后向下进行处理。

（1）应用层

应用层是网络应用程序及它们的应用层协议存留的地方。因特网的应用层包括许多协议，例如HTTP（它提供了Web文档的请求和传送）、SMTP（它提供了电子邮件报文的传输）和FTP（它提供两个端系统之间的文件传送）。我们将看到，某些网络功能，如将像www.ietf.org这样对人友好的端系统名字转换为32比特的网络地址，也是借助于特定的应用层协议即域名系统（DNS）完成的。我们将在第2章中看到，创建并部署我们自己的新应用层协议是非常容易的。



a）5层因特网协议栈　　b）7层ISO OSI参考模型

图1-23　因特网协议栈和OSI参考模型

应用层协议分布在多个端系统上，而一个端系统中的应用程序使用协议与另一个端系统中的应用程序交换信息分组。我们把这种位于应用层的信息分组称为**报文**（message）。

（2）运输层

因特网的运输层在应用程序端点之间传送应用层报文。在因特网中，有两种运输协议，即TCP和UDP，利用其中的任一个都能运输应用层报文。TCP向它的应用程序提供了面向连接的服务。这种服务包括了应用层报文向目的地的确保传递和流量控制（即发送方/接收方速率匹配）。TCP也将长报文划分为短报文，并提供拥塞控制机制，因此当网络拥塞时，源抑制其传输速率。UDP协议向它的应用程序提供无连接服务。这是一种不提供不必要服务的服务，没有可靠性，没有流量控制，也没有拥塞控制。在本书中，我们把运输层的分组称为**报文段**（segment）。

（3）网络层

因特网的网络层负责将称为**数据报**（datagram）的网络层分组从一台主机移动到另一台主机。在一台源主机中的因特网运输层协议（TCP或UDP）向网络层递交运输层报文段和目的地址，就像你通过邮政服务寄信件时提供一个目的地址一样。

因特网的网络层包括著名的网际协议IP，该协议定义了在数据报中的各个字段以及端系统和路由器如何作用于这些字段。IP仅有一个，所有具有网络层的因特网组件必须运行IP。因特网的网络层也包括决定路由的路由选择协议，它根据该路由将数据报从源传输到目的地。因特网具有许多路由选择协议。如我们在1.3节所见，因特网是一个网络的网络，并且在一个网络中，其网络管理者能够运行所希望的任何路由选择协议。尽管网络层包括了网际协议和一些路由选择协议，但通常把它简单地称为IP层，这反映了IP是将因特网连接在一起的黏合剂这样的事实。

（4）链路层

因特网的网络层通过源和目的地之间的一系列路由器路由数据报。为了将分组从一个节点（主机或路由器）移动到路径上的下一个节点，网络层必须依靠该链路层的服务。特别是在每个节点，网络层将数据报下传给链路层，链路层沿着路径将数据报传递给下一个节点。在该下一个节点，链路层将数据报上传给网络层。

由链路层提供的服务取决于应用于该链路的特定链路层协议。例如，某些协议基于链路提供可靠传递，从传输节点跨越一条链路到接收节点。值得注意的是，这种可靠的传递服务不同于TCP的可靠传递服务，TCP提供从一个端系统到另一个端系统的可靠交付。链路层的例子包括以太网、WiFi和电缆接入网的DOCSIS协议。因为数据报从源到目的地传送通常需要经过几条链路，一个数据报可能被沿途不同链路上的不同链路层协议处理。例如，一个数据报可能被一段链路上的以太网和下一段链路上的PPP所处理。网络层将受到来自每个不同的链路层协议的不同服务。在本书中，我们把链路层分组称为**帧**（frame）。

（5）物理层

虽然链路层的任务是将整个帧从一个网络元素移动到邻近的网络元素，而物理层的任务是将该帧中的一个个比特从一个节点移动到下一个节点。在这层中的协议仍然是链路相关的，并且进一步与该链路（例如，双绞铜线、单模光纤）的实际传输媒体相关。例如，以太网具有许多物理层协议：一个是关于双绞铜线的，另一个是关于同轴电缆的，还有一个是关于光纤的，等等。在每种场合中，跨越这些链路移动一个比特是以不同的方式进行的。

2. OSI模型

详细地讨论过因特网协议栈后，我们应当提及它不是唯一的协议栈。特别是在20世纪70年代后期，国际标准化组织（ISO）提出计算机网络围绕7层来组织，称为开放系统

互连（OSI）模型［ISO 2016］。当那些要成为因特网协议的协议还处于襁褓之中，只是许多正在研发的不同协议族之一时，OSI 模型就已经成形。事实上，初始 OSI 模型的发明者在创建该模型时心中可能并没有想到因特网。无论如何，自 20 世纪 70 年代后期开始，许多培训课程和大学课程都围绕 7 层模型挑选有关 ISO 授权和组织的课程。因为它在网络教育的早期影响，该 7 层模型继续以某种方式存留在某些网络教科书和培训课程中。

显示在图 1-23b 中的 OSI 参考模型的 7 层是：应用层、表示层、会话层、运输层、网络层、数据链路层和物理层。这些层次中，5 层的功能大致与它们名字类似的因特网对应层的功能相同。所以，我们来考虑 OSI 参考模型中附加的两个层，即表示层和会话层。表示层的作用是使通信的应用程序能够解释交换数据的含义。这些服务包括数据压缩和数据加密（它们是自解释的）以及数据描述（这使得应用程序不必担心在各台计算机中表示/存储的内部格式不同的问题）。会话层提供了数据交换的定界和同步功能，包括了建立检查点和恢复方案的方法。

因特网缺少了在 OSI 参考模型中建立的两个层次，该事实引起了一些有趣的问题：这些层次提供的服务不重要吗？如果一个应用程序需要这些服务之一，将会怎样呢？因特网对这两个问题的回答是相同的：这留给应用程序开发者处理。应用程序开发者决定一个服务是否是重要的，如果该服务重要，应用程序开发者就应该在应用程序中构建该功能。

1.5.2 封装

图 1-24 显示了这样一条物理路径：数据从发送端系统的协议栈向下，沿着中间的链路层交换机和路由器的协议栈上上下下，然后向上到达接收端系统的协议栈。如我们将在本书后面讨论的那样，路由器和链路层交换机都是分组交换机。与端系统类似，路由器和链路层交换机以多层次的方式组织它们的网络硬件和软件。而路由器和链路层交换机并不实现协议栈中的所有层次。如图 1-24 所示，链路层交换机实现了第一层和第二层；路由器实现了第一层到第三层。例如，这意味着因特网路由器能够实现 IP 协议（一种第三层协议），而链路层交换机则不能。我们将在后面看到，尽管链路层交换机不能识别 IP 地址，但它们能够识别第二层地址，如以太网地址。值得注意的是，主机实现了所有 5 个层次，这与因特网体系结构将它的复杂性放在网络边缘的观点是一致的。

图 1-24 也说明了一个重要概念：**封装**（encapsulation）。在发送主机端，一个**应用层报文**（application-layer message）（图 1-24 中的 M）被传送给运输层。在最简单的情况下，运输层收取到报文并附上附加信息（所谓运输层首部信息，图 1-24 中的 H_t），该首部将被接收端的运输层使用。应用层报文和运输层首部信息一道构成了**运输层报文段**（transport-layer segment）。运输层报文段因此封装了应用层报文。附加的信息也许包括了下列信息：允许接收端运输层向上向适当的应用程序交付报文的信息；差错检测位信息，该信息让接收方能够判断报文中的比特是否在途中已被改变。运输层则向网络层传递该报文段，网络层增加了如源和目的端系统地址等网络层首部信息（图 1-24 中的 H_n），生成了**网络层数据报**（network-layer datagram）。该数据报接下来被传递给链路层，链路层（自然而然地）增加它自己的链路层首部信息并生成**链路层帧**（link-layer frame）。所以我们看到，在每一层，一个分组具有两种类型的字段：首部字段和**有效载荷字段**（payload field）。有效载荷通常是来自上一层的分组。

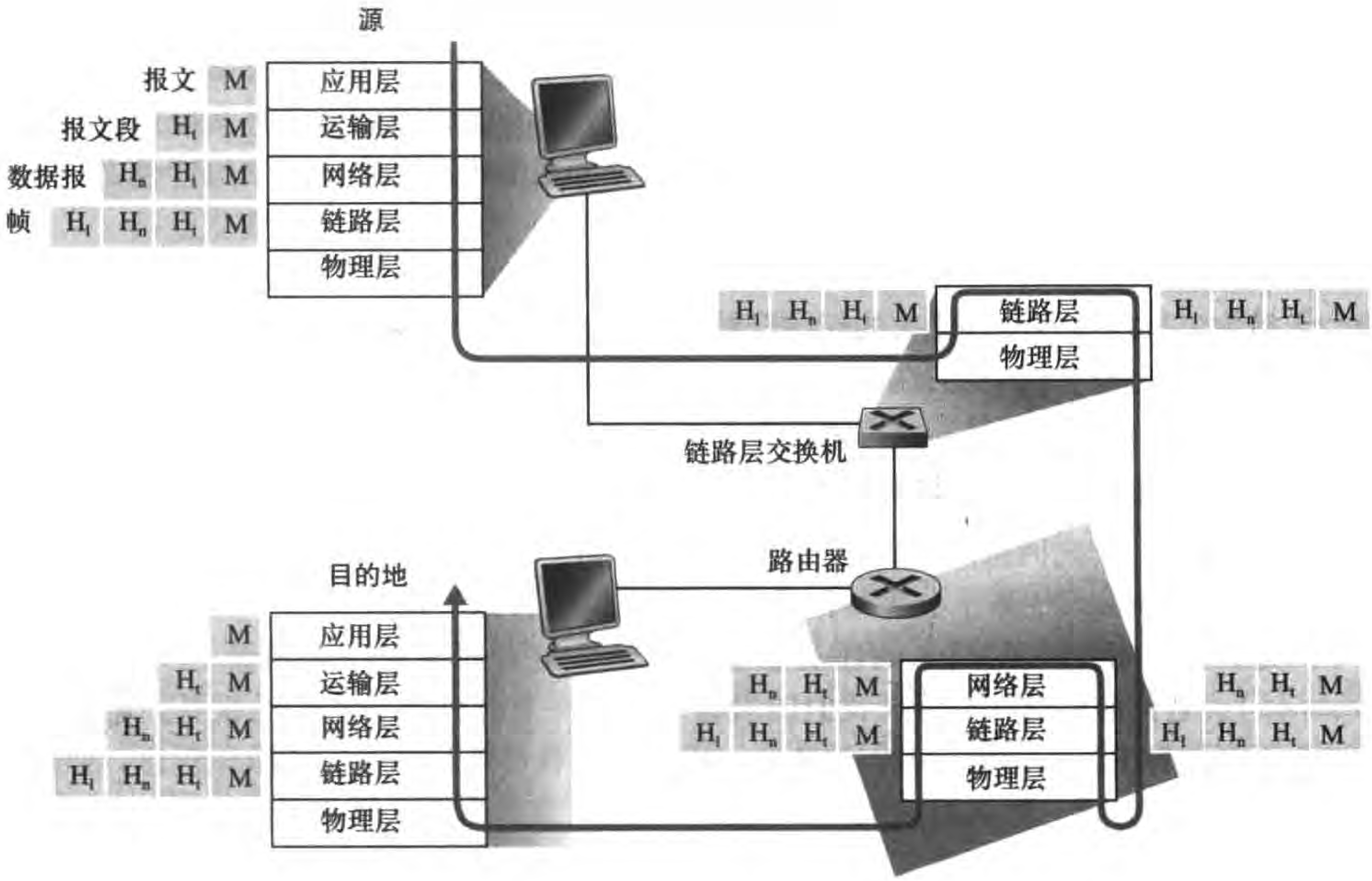


图 1-24 主机、路由器和链路层交换机，每个包含了不同的层，反映了它们的功能差异

这里一个有用的类比是经过公共邮政服务在某公司办事处之间发送一封备忘录。假定位于某办事处的 Alice 要向位于另一办事处的 Bob 发送一封备忘录。该备忘录类比于应用层报文。Alice 将备忘录放入办事处之间的公函信封中，并在公函信封上方写上了 Bob 的名字和部门。该办事处之间的公函信封类比于运输层报文段，即它包括了首部信息（Bob 的名字和部门编号）并封装了应用层报文（备忘录）。当发送办事处的收发室拿到该办事处之间的备忘录时，将其放入适合在公共邮政服务发送的信封中，并在邮政信封上写上发送和接收办事处的邮政地址。此处，邮政信封类比于数据报，它封装了运输层的报文段（办事处之间的公函信封），该报文段封装了初始报文（备忘录）。邮政服务将该邮政信封交付给接收办事处的收发室。在此处开始了拆封过程。该收发室取出了办事处之间的公函信封并转发给 Bob。最后，Bob 打开信封并拿走了备忘录。

封装的过程能够比前面描述的更为复杂。例如，一个大报文可能被划分为多个运输层的报文段（这些报文段每个可能被划分为多个网络层数据报）。在接收端，则必须从其连续的数据报中重构这样一个报文段。

1.6 面对攻击的网络

对于今天的许多机构（包括大大小小的公司、大学和政府机关）而言，因特网已经成为与其使命密切相关的一部分了。许多人也依赖因特网从事各种职业、社会和个人活动。目前，数以亿计的物品（包括可穿戴设备和家用设备）与因特网相连。但是在所有这一切背后，存在着一个阴暗面，其中的“坏家伙”试图对我们的日常生活进行破坏，如损坏我们与因特网相连的计算机，侵犯我们的隐私以及使我们依赖的因特网服务无法运行。

网络安全领域主要探讨以下问题：坏家伙如何攻击计算机网络，以及我们（即将成为计算机网络的专家）如何防御以免受他们的攻击，或者更好的是设计能够事先免除这样的攻击的新型体系结构。面对经常发生的各种各样的现有攻击以及新型和更具摧毁性的未来攻击的威胁，网络安全已经成为近年来计算机网络领域的中心主题。本书的特色之一是将网络安全问题放在中心位置。

因为我们在计算机网络和因特网协议方面还没有专业知识，所以这里我们将从审视某些今天最为流行的与安全性相关的问题开始。这将刺激我们的胃口，以便我们在后续章节中进行更为充实的讨论。我们在这里以提出问题开始：什么会出现问题？计算机网络是如何受到攻击的？今天一些最为流行的攻击类型是什么？

1. 坏家伙能够经因特网将有害程序放入你的计算机中

因为我们要从/向因特网接收/发送数据，所以我们将设备与因特网相连。这包括各种好东西，例如 Instagram 帖子、因特网搜索结果、流式音乐、视频会议、流式电影等。但不幸的是，伴随好的东西而来的还有恶意的东西，这些恶意的东西可统称为**恶意软件**（malware），它们能够进入并感染我们的设备。一旦恶意软件感染我们的设备，就能够做各种不正当的事情，包括删除我们的文件，安装间谍软件来收集我们的隐私信息，如社会保险号、口令和击键，然后将这些（当然经因特网）发送给坏家伙。我们的受害主机也可能成为数以千计的类似受害设备网络中的一员，它们被统称为**僵尸网络**（botnet），坏家伙利用僵尸网络控制并有效地对目标主机展开垃圾邮件分发或分布式拒绝服务攻击（很快将讨论）。

至今为止的多数恶意软件是**自我复制**（self-replicating）的：一旦它感染了一台主机，就会从那台主机寻求进入因特网上的其他主机，从而形成新的感染主机，再寻求进入更多的主机。以这种方式，自我复制的恶意软件能够指数式地快速扩散。恶意软件能够以病毒或蠕虫的形式扩散。**病毒**（virus）是一种需要某种形式的用户交互来感染用户设备的恶意软件。典型的例子是包含恶意可执行代码的电子邮件附件。如果用户接收并打开这样的附件，不经意间就在其设备上运行了该恶意软件。通常，这种电子邮件病毒是自我复制的：例如，一旦执行，该病毒可能向用户地址簿上的每个接收方发送一个具有相同恶意附件的相同报文。**蠕虫**（worm）是一种无须任何明显用户交互就能进入设备的恶意软件。例如，用户也许运行了一个攻击者能够发送恶意软件的脆弱网络应用程序。在某些情况下，没有用户的任何干预，该应用程序可能从因特网接收恶意软件并运行它，生成了蠕虫。新近感染设备中的蠕虫则能扫描因特网，搜索其他运行相同网络应用程序的易受感染的主机。当它发现其他易受感染的主机时，便向这些主机发送一个它自身的副本。今天，恶意软件无所不在且防范成本高。当你用这本书学习时，我们鼓励你思考下列问题：计算机网络设计者能够采取什么防御措施，以使与因特网连接的设备免受恶意软件的攻击？

2. 坏家伙能够攻击服务器和网络基础设施

另一种宽泛类型的安全性威胁称为**拒绝服务攻击**（Denial-of-Service（DoS）attack）。顾名思义，DoS 攻击使得网络、主机或其他基础设施部分不能由合法用户使用。Web 服务器、电子邮件服务器、DNS 服务器（在第 2 章中讨论）和机构网络都能够成为 DoS 攻击的目标。因特网 DoS 攻击极为常见，每年会出现数以千计的 DoS 攻击 [Moore 2001]。访问数字攻击图（Digital Attack Map）站点可以观看世界范围内每天最厉害的 DoS 攻击 [DAM 2016]。大多数因特网 DoS 攻击属于下列三种类型之一：

- *弱点攻击*。这涉及向一台目标主机上运行的易受攻击的应用程序或操作系统发送制作精细的报文。如果适当顺序的多个分组发送给一个易受攻击的应用程序或操作系统，该服务器可能停止运行，或者更糟糕的是主机可能崩溃。
- *带宽洪泛*。攻击者向目标主机发送大量的分组，分组数量之多使得目标的接入链路变得拥塞，使得合法的分组无法到达服务器。
- *连接洪泛*。攻击者在目标主机中创建大量的半开或全开 TCP 连接（将在第 3 章中讨论 TCP 连接）。该主机因这些伪造的连接而陷入困境，并停止接受合法的连接。

我们现在更详细地研究这种带宽洪泛攻击。回顾 1.4.2 节中讨论的时延和丢包问题，显然，如果某服务器的接入速率为 R bps，则攻击者将需要以大约 R bps 的速率来产生危害。如果 R 非常大的话，单一攻击源可能无法产生足够大的流量来伤害该服务器。此外，如果从单一源发出所有流量的话，某上游路由器就能够检测出该攻击并在该流量靠近服务器之前就将其阻挡下来。在图 1-25 中显示的**分布式 DoS**（Distributed DoS，DDoS）中，攻击者控制多个源并让每个源向目标猛烈发送流量。使用这种方法，遍及所有受控源的聚合流量速率需要大约 R 的能力来使该服务陷入瘫痪。DDoS 攻击充分利用由数以千计的受害主机组成的僵尸网络，这在今天是屡见不鲜的［DAM 2016］。相比于来自单一主机的 DoS 攻击，DDoS 攻击更加难以检测和防范。

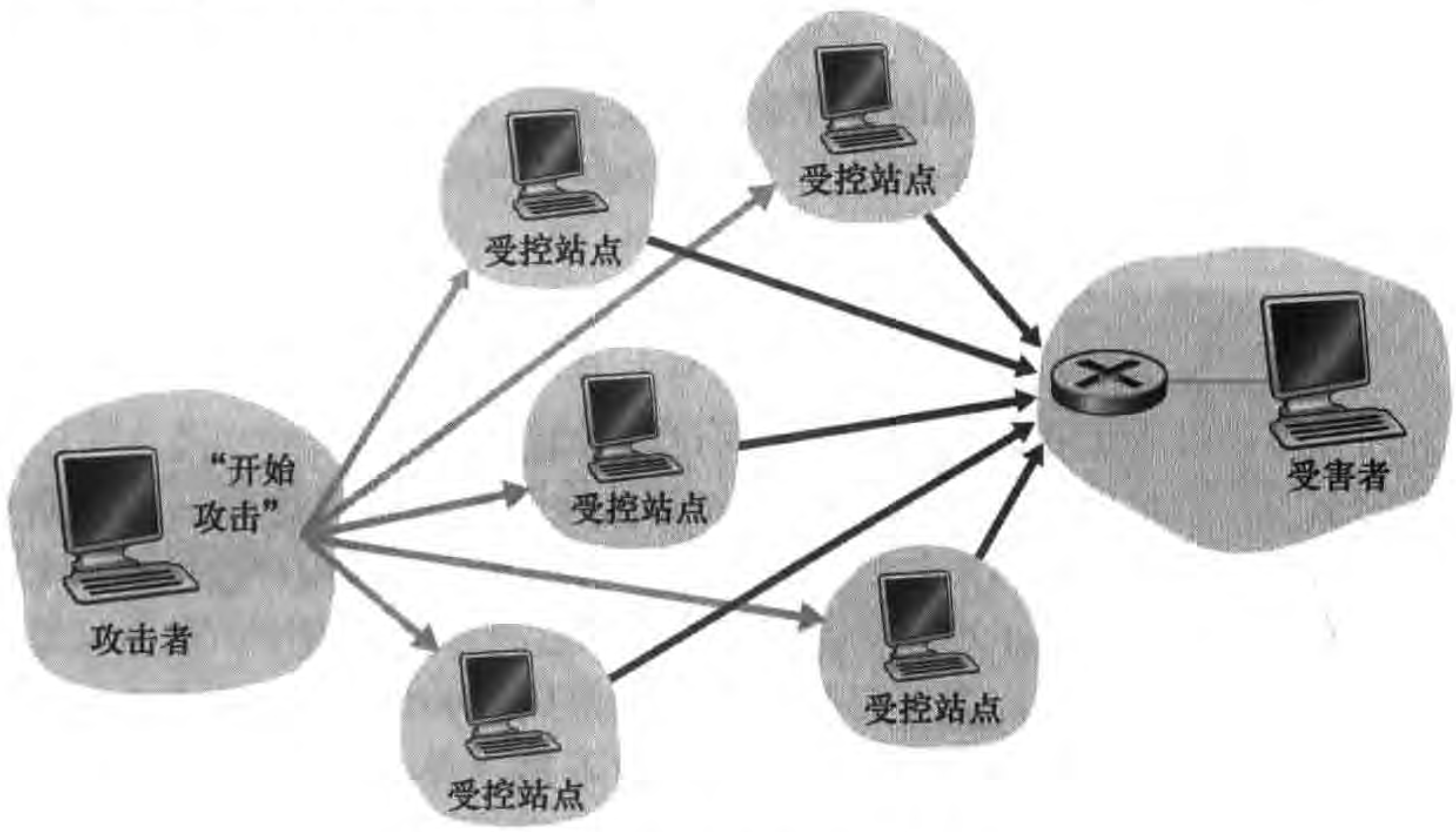


图 1-25　分布式拒绝服务攻击

当学习这本书时，我们鼓励你考虑下列问题：计算机网络设计者能够采取哪些措施防止 DoS 攻击？我们将看到，对于 3 种不同类型的 DoS 攻击需要采用不同的防御方法。

3. 坏家伙能够嗅探分组

今天的许多用户经无线设备接入因特网，如 WiFi 连接的膝上计算机或使用蜂窝因特网连接的手持设备（在第 7 章中讨论）。无所不在的因特网接入极为便利并让移动用户方便地使用令人惊奇的新应用程序的同时，也产生了严重的安全脆弱性——在无线传输设备的附近放置一台被动的接收机，该接收机就能得到传输的每个分组的副本！这些分组包含了各种敏感信息，包括口令、社会保险号、商业秘密和隐秘的个人信息。记录每个流经的分组副本的被动接收机被称为**分组嗅探器**（packet sniffer）。

嗅探器也能够部署在有线环境中。在有线的广播环境中，如在许多以太网 LAN 中，

分组嗅探器能够获得经该 LAN 发送的所有分组。如在 1.2 节中描述的那样，电缆接入技术也广播分组，因此易于受到嗅探攻击。此外，获得某机构与因特网连接的接入路由器或接入链路访问权的坏家伙能够放置一台嗅探器以产生从该机构出入的每个分组的副本，再对嗅探到的分组进行离线分析，就能得出敏感信息。

分组嗅探软件在各种 Web 站点上可免费得到，这类软件也有商用的产品。教网络课程的教授布置的实验作业就涉及写一个分组嗅探器和应用层数据重构程序。与本书相关联的 Wireshark [Wireshark 2016] 实验（参见本章结尾处的 Wireshark 实验介绍）使用的正是这样一种分组嗅探器！

因为分组嗅探器是被动的，也就是说它们不向信道中注入分组，所以难以检测到它们。因此，当我们向无线信道发送分组时，我们必须接受这样的可能性，即某些坏家伙可能记录了我们的分组的副本。如你已经猜想的那样，最好的防御嗅探的方法基本上都与密码学有关。我们将在第 8 章研究密码学应用于网络安全的有关内容。

4. 坏家伙能够伪装成你信任的人

生成具有任意源地址、分组内容和目的地址的分组，然后将这个人工制作的分组传输到因特网中，因特网将忠实地将该分组转发到目的地，这一切都极为容易（当你学完这本教科书后，你将很快具有这方面的知识了！）。想象某个接收到这样一个分组的不会猜疑的接收方（比如说一台因特网路由器），将该（虚假的）源地址作为真实的，进而执行某些嵌入在该分组内容中的命令（比如说修改它的转发表）。将具有虚假源地址的分组注入因特网的能力被称为 **IP 哄骗**（IP spoofing），而它只是一个用户能够冒充另一个用户的许多方式中的一种。

为了解决这个问题，我们需要采用*端点鉴别*，即一种使我们能够确信一个报文源自我们认为它应当来自的地方的机制。当你继续学习本书各章时，再次建议你思考怎样为网络应用程序和协议做这件事。我们将在第 8 章探讨端点鉴别机制。

在本节结束时，值得思考一下因特网是如何从一开始就落入这样一种不安全的境地的。大体上讲，答案是：因特网最初就是基于“一群相互信任的用户连接到一个透明的网络上”这样的模型 [Blumenthal 2001] 进行设计的，在这样的模型中，安全性是没有必要的。初始的因特网体系结构在许多方面都深刻地反映了这种相互信任的理念。例如，一个用户向任何其他用户发送分组的能力是默认的，而不是一种请求/准予的能力，还有用户身份取自所宣称的表面价值，而不是默认地需要鉴别。

但是今天的因特网无疑并不涉及“相互信任的用户”。但是，今天的用户仍然需要通信，当他们不必相互信任时，他们也许希望匿名通信，也许间接地通过第三方通信（例如我们将在第 2 章中学习的 Web 高速缓存，我们将在第 7 章学习的移动性协助代理），也许不信任他们通信时使用的硬件、软件甚至空气。随着我们进一步学习本书，会面临许多安全性相关的挑战：我们应当寻求对嗅探、端点假冒、中间人攻击、DDoS 攻击、恶意软件等的防护办法。我们应当记住：在相互信任的用户之间的通信是一种例外而不是规则。欢迎你到现代计算机网络世界！

1.7 计算机网络和因特网的历史

1.1 节到 1.6 节概述了计算机网络和因特网的技术。你现在应当有足够的知识来给家人和朋友留下深刻印象了。然而，如果你真的想在下次鸡尾酒会上一鸣惊人，你应当在你

的演讲中点缀一些有关因特网引人入胜的历史轶闻 [Segaller 1998]。

1.7.1 分组交换的发展：1961~1972

计算机网络和今天因特网领域的开端可以追溯到20世纪60年代早期，那时电话网是世界上占统治地位的通信网络。1.3节讲过，电话网使用电路交换将信息从发送方传输到接收方，这种适当的选择使得语音以一种恒定的速率在发送方和接收方之间传输。随着20世纪60年代早期计算机的重要性越来越大，以及分时计算机的出现，考虑如何将计算机连接在一起，并使它们能够被地理上分布的用户所共享的问题，也许就成了一件自然的事。这些用户所产生的流量很可能具有*突发性*，即活动的间断性，例如向远程计算机发送一个命令，接着是静止的时间段，这是等待应答或对接收到的响应进行思考的时间。

全世界有3个研究组首先发明了分组交换，以作为电路交换的一种有效的、健壮的替代技术。这3个研究组互不知道其他人的工作 [Leiner 1998]。有关分组交换技术的首次公开发表出自 Leonard Kleinrock [Kleinrock 1961; Kleinrock 1964]，那时他是麻省理工学院（MIT）的一名研究生。Kleinrock 使用排队论，完美地体现了使用分组交换方法处理突发性流量源的有效性。1964年，兰德公司的 Paul Baran [Baran 1964] 已经开始研究分组交换的应用，以在军用网络上传输安全语音，同时在英国的国家物理实验室（NPL），Donald Davies 和 Roger Scantlebury 也在研究分组交换技术。

MIT、兰德和 NPL 的工作奠定了今天的因特网的基础。但是因特网也经历了很长的"边构建边示范（let's-build-it-and-demonstrate-it）"的历史，这可追溯到20世纪60年代早期。J. C. R. Licklider [DEC 1990] 和 Lawrence Roberts 都是 Kleinrock 在 MIT 的同事，他们转而去领导美国高级研究计划署（Advanced Research Projects Agency，ARPA）的计算机科学计划。Roberts 公布了一个 ARPAnet [Roberts 1967] 的总体计划，它是第一个分组交换计算机网络，是今天的公共因特网的直接祖先。在1969年的劳动节，第一台分组交换机在 Kleinrock 的监管下安装在美国加州大学洛杉矶分校（UCLA），其他3台分组交换机不久后安装在斯坦福研究所（Stanford Research Institute，SRI）、美国加州大学圣巴巴拉分校（UC Santa Barbara）和犹他大学（University of Utah）（参见图1-26）。羽翼未丰的因特网祖先到1969年年底有了4个节点。Kleinrock 回忆说，该网络的最先应用是从 UCLA 到 SRI 执行远程注册，但却导致了该系统的崩溃 [Kleinrock 2004]。



图1-26 一台早期的分组交换机

到了1972年，ARPAnet 已经成长到大约15个节点，由 Robert Kahn 首次对它进行了公开演示。在 ARPAnet 端系统之间的第一台主机到主机协议——称为网络控制协议

(NCP), 就是此时完成的 [RFC 001]。随着端到端协议的可供使用, 这时能够写应用程序了。在 1972 年, Ray Tomlinson 编写了第一个电子邮件程序。

1.7.2 专用网络和网络互联: 1972 ~ 1980

最初的 ARPAnet 是一个单一的、封闭的网络。为了与 ARPAnet 的一台主机通信, 一台主机必须与另一台 ARPAnet IMP 实际相连。20 世纪 70 年代早期和中期, 除 ARPAnet 之外的其他分组交换网络问世: ALOHAnet 是一个微波网络, 它将夏威夷岛上的大学 [Abramson 1970] 以及 DARPA 的分组卫星 [RFC 829] 和分组无线电网 [Kahn 1987] 连接到一起; Telenet 是 BBN 的商用分组交换网, 它基于 ARPAnet 技术; 由 Louis Pouzin 领衔的 Cyclades 是法国的一个分组交换网 [Think 2012]; 还有如 Tymnet 和 GE 信息服务网这样的分时网络, 以及 20 世纪 60 年代后期和 70 年代初期的类似网络 [Schwartz 1977]; IBM 的 SNA (1996 ~ 1974), 它与 ARPAnet 同时在运行 [Schwartz 1977]。

网络的数目开始增加。人们事后看到, 研制将网络连接到一起的体系结构的时机已经成熟。互联网络的先驱性工作 (得到了美国国防部高级研究计划署 (DARPA) 的支持) 由 Vinton Cerf 和 Robert Kahn [Cerf 1974] 完成, 本质上就是创建一个网络的网络; 术语*网络互联* (internetting) 就是用来描述该项工作的。

这些体系结构的原则体现在 TCP 中。然而, TCP 的早期版本与今天的 TCP 差异很大。TCP 的早期版本将通过端系统重传的可靠按序数据传递 (仍是今天的 TCP 的一部分) 与转发功能 (今天该功能由 IP 执行) 相结合。TCP 的早期实验以及认识到对诸如分组语音这样的应用程序中不可靠的、非流控制的、端到端传递服务的重要性, 导致 IP 从 TCP 中分离出来, 并研制了 UDP 协议。我们今天看到的 3 个重要的因特网协议——TCP、UDP 和 IP, 到 20 世纪 70 年代末在概念上已经完成。

除了 DARPA 的因特网相关研究外, 许多其他重要的网络活动也在进行中。在夏威夷, Norman Abramson 正在研制 ALOHAnet, 这是一个基于分组的无线电网络, 它使在夏威夷岛上的多个远程站点互相通信。ALOHA 协议 [Abramson 1970] 是第一个多路访问协议, 允许地理上分布的用户共享单一的广播通信媒体 (一个无线电频率)。Metcalfe 和 Boggs 基于 Abramson 的多路访问协议, 研制了用于有线共享广播网络的以太网协议 [Metcalfe 1976]。令人感兴趣的是, Metcalfe 和 Boggs 的以太网协议是由连接多台 PC、打印机和共享磁盘在一起的需求所激励的 [Perkins 1994]。在 PC 革命和网络爆炸的 25 年之前, Metcalfe 和 Boggs 就奠定了今天 PC LAN 的基础。

1.7.3 网络的激增: 1980 ~ 1990

到了 20 世纪 70 年代末, 大约 200 台主机与 ARPAnet 相连。到了 20 世纪 80 年代末, 连到公共因特网的主机数量达到 100 000 台, 那时的公共因特网是网络的联盟, 看起来非常像今天的因特网。20 世纪 80 年代是联网主机数量急剧增长的时期。

这种增长是由几个显著成果即创建计算机网络将大学连接到一起引起的。BITNET 为位于美国东北部的几个大学之间提供了电子邮件和文件传输。建立了 CSNET (计算机科学网), 以将还没有接入 ARPAnet 的大学研究人员连接在一起。1986 年, 建立了 NSFNET, 为 NSF 资助的超级计算中心提供接入。NSFNET 最初具有 56kbps 的主干速率, 到了 20 世纪 80 年代末, 它的主干运行速率是 1.5Mbps, 并成为连接区域网络的基本主干。

在 ARPAnet 界中, 许多今天的因特网体系结构的最终部分逐渐变得清晰起来。1983

年1月1日见证了TCP/IP作为ARPAnet新的标准主机协议的正式部署，替代了NCP协议。从NCP到TCP/IP的迁移［RFC 801］是一个标志性事件，所有主机被要求在那天转移到TCP/IP上去。在20世纪80年代后期，TCP进行了重要扩展，以实现基于主机的拥塞控制［Jacobson 1988］。还研制出了DNS（域名系统），用于将人可读的因特网名字（例如gaia.cs.umass.edu）映射到它的32比特IP地址［RFC 1034］。

在20世纪80年代初期，在ARPAnet（这绝大多数是美国的成果）发展的同时，法国启动了Minitel项目，这个雄心勃勃的计划是让数据网络进入每个家庭。在法国政府的支持下，Minitel系统由公共分组交换网络（基于X.25协议集）、Minitel服务器和具有内置低速调制解调器的廉价终端组成。Minitel于1984年取得了巨大的成功，当时法国政府向每个需要的住户免费分发一个Minitel终端。Minitel站点包括免费站点（如电话目录站点）以及一些专用站点，这些专用站点根据每个用户的使用来收取费用。在20世纪90年代中期的鼎盛时期，Minitel提供了20 000多种服务，涵盖从家庭银行到特殊研究数据库的广泛范围。Minitel在大量法国家庭中存在了10年后，大多数美国人才听说因特网。

1.7.4 因特网爆炸：20世纪90年代

20世纪90年代出现了许多事件，这些事件标志着因特网持续革命和很快到来的商业化。作为因特网祖先的ARPAnet已不复存在。1991年，NSFNET解除了对NSFNET用于商业目的的限制。NSFNET自身于1995年退役，这时因特网主干流量则由商业因特网服务提供商负责承载。

然而，20世纪90年代的主要事件是万维网（World Wide Web）应用程序的出现，它将因特网带入世界上数以百万计的家庭和商业中。Web作为一个平台，也引入和配置了数百个新的应用程序，其中包括搜索（如谷歌和Bing）、因特网商务（如亚马逊和eBay）以及社交网络（如脸书），对这些应用程序我们今天已经习以为常了。

Web是由Tim Berners-Lee于1989～1991年间在CERN发明的［Berners-Lee 1989］，最初的想法源于20世纪40年代Vannevar Bush［Bush 1945］和20世纪60年代以来Ted Nelson［Xanadu 2012］在超文本方面的早期工作。Berners-Lee和他的同事研制了HTML、HTTP、Web服务器和浏览器的初始版本，这是Web的4个关键部分。到了1993年年底前后，大约有200台Web服务器在运行，而这些只是正在出现的Web服务器的冰山一角。就在这个时候，几个研究人员研制了具有GUI接口的Web浏览器，其中的Marc Andreessen和Jim Clark一起创办了Mosaic Communications公司，该公司就是后来的Netscape通信公司［Cusmano 1998；Quittner 1998］。到了1995年，大学生们每天都在使用Netscape浏览器在Web上冲浪。大约在这段时间，大大小小的公司都开始运行Web服务器，并在Web上处理商务。1996年，微软公司开始开发浏览器，这导致了Netscape和微软之间的浏览器之战，并以微软公司在几年后获胜而告终［Cusumano 1998］。

20世纪90年代的后5年，随着主流公司和数以千计的初创公司创造了大量因特网产品和服务，因特网到了飞速增长和创新的时期。到了2000年末，因特网已经支持数百流行的应用程序，包括以下4种备受欢迎的应用程序：

- 电子邮件，包括附件和Web可访问的电子邮件。
- Web，包括Web浏览和因特网商务。
- 即时讯息（instant messaging），具有联系人列表。
- MP3的对等（peer-to-peer）文件共享，由Napster开创。

值得一提的是，前两个应用程序出自专业研究机构，而后两个却由一些年轻创业者所发明。

1995~2001年，这段时间也是因特网在金融市场上急转突变的时期。在成为有利可图的公司之前，数以百计的因特网初创公司靠首次公开募股（IPO）并在股票市场上交易起家。许多公司身价数十亿美元，却没有任何主要的收入渠道。因特网的股票在2000~2001年崩盘，导致许多初创公司倒闭。不过，也有许多公司成为因特网世界的大赢家，包括微软、思科、雅虎、e-Bay、谷歌和亚马逊。

1.7.5 最新发展

计算机网络中的变革继续以急促的步伐前进。所有的前沿研究正在取得进展，包括部署更快的路由器和在接入网和网络主干中提供更高的传输速率。但下列进展值得特别关注：

- 自2000年开始，我们见证了家庭宽带因特网接入的积极部署——不仅有电缆调制解调器和DSL，而且有光纤到户，这些在1.2节中讨论过。这种高速因特网为丰富的视频应用创造了条件，包括用户生成的视频的分发（例如YouTube）、电影和电视节目的按需流（例如Netflix）以及多人视频会议（例如Skype、Facetime和Google Hangouts）。
- 高速（54Mbps及更高）公共WiFi网络和经过4G蜂窝电话网的中速（几十Mbps）因特网接入越来越普及，不仅使在运动中保持持续连接成为可能，也产生了新型特定位置应用，如Yelp、Tinder、Yik Yak和Waz。2011年，与因特网连接的无线设备的数量超过了有线设备的数量。高速无线接入为手持计算机（iPhone、安卓手机、iPad等）的迅速出现提供了舞台，这些手持计算机具有对因特网持续不断和无拘束接入的优点。
- 诸如脸书、Instagram、推特（Twitter）和微信（在中国极为流行）这样的在线社交网络已经在因特网之上构建了巨大的人际网络。这些社交网络，许多广泛用于发送消息以及照片分享。许多因特网用户今天主要“生活”在一个或多个社交网络中。通过他们的API，在线社交网络为新的联网应用和分布式游戏创建了平台。
- 如在1.3.3节中所讨论的，在线服务提供商如谷歌和微软已经广泛部署了自己的专用网络。该专用网络不仅将它们分布在全球的数据中心连接在一起，而且通过直接与较低层ISP对等连接，能够尽可能绕过因特网。因此，谷歌几乎可以瞬间提供搜索结果和电子邮件访问，仿佛它们的数据中心运行在自己的计算机中一样。
- 许多因特网商务公司在“云”（如亚马逊的EC2、谷歌的应用引擎、微软的Azure）中运行它们的应用。许多公司和大学也已经将它们的因特网应用（如电子邮件和Web托管）迁移到云中。云公司不仅可以为应用提供可扩展的计算和存储环境，也可为应用提供对其高性能专用网络的隐含访问。

1.8 小结

在本章中，我们涉及了大量的材料！我们已经看到构成特别的因特网以及一般的计算机网络的各种硬件和软件。我们从网络的边缘开始，观察端系统和应用程序，以及运行在

端系统上为应用程序提供的运输服务。接着我们也观察了通常能够在接入网中找到的链路层技术和物理媒体。然后我们进入网络核心更深入地钻研网络，看到分组交换和电路交换是通过电信网络传输数据的两种基本方法，并且探讨了每种方法的长处和短处。我们也研究了全球性因特网的结构，知道了因特网是网络的网络。我们看到了因特网的由较高层和较低层 ISP 组成的等级结构，允许该网络扩展为包括数以千计的网络。

在这个概述性章节的第二部分，我们研究了计算机网络领域的几个重要主题。我们首先研究了分组交换网中的时延、吞吐量和丢包的原因。我们研究得到传输、传播和排队时延以及用于吞吐量的简单定量模型；我们将在整本书的课后习题中多处使用这些时延模型。接下来，我们研究了协议分层和服务模型、网络中的关键体系结构原则，我们将在本书多处引用它们。我们还概述了在今天的因特网中某些更为流行的安全攻击。我们用计算机网络的简要历史结束我们对网络的概述。第 1 章本身就构成了计算机网络的小型课程。

因此，第 1 章中的确涉及了大量的背景知识！如果你有些不知所云，请不要着急。在后继几章中我们将重新回顾这些概念，更为详细地研究它们（那是承诺，而不是威胁!）。此时，我们希望你完成本章内容的学习时，对构建网络的众多元素的直觉越来越敏锐，对网络词汇越来越精通（不妨经常回过头来查阅本章），对更加深入地学习网络的愿望越来越强烈。这些也是在本书的其余部分我们将面临的任务。

本书的路线图

在开始任何旅行之前，你总要先查看路线图，以便更为熟悉前面的主要道路和交叉路口。对于我们即将开启的这段“旅行”而言，其最终目的地是深入理解计算机网络“是什么、怎么样和为什么”等内容。我们的路线图是本书各章的顺序：

第 1 章　计算机网络和因特网
第 2 章　应用层
第 3 章　运输层
第 4 章　网络层：数据平面
第 5 章　网络层：控制平面
第 6 章　链路层和局域网
第 7 章　无线网络和移动网络
第 8 章　计算机网络中的安全
第 9 章　多媒体网络

第 2 ~ 6 章是本书的 5 个核心章。应当注意的是，这些章都围绕 5 层因特网协议栈上面的 4 层而组织，其中一章对应一层。要进一步注意的是，我们的旅行将从因特网协议栈的顶部即应用层开始，然后向下面各层进行学习。这种自顶向下旅行背后的基本原理是，一旦我们理解这些应用程序，就能够理解支持这些应用程序所需的各种网络服务。然后能够依次研究可能由网络体系结构实现的服务的各种方式。较早地涉及应用程序，也能够对学习本课程其余部分提供动力。

第 7 ~ 9 章关注现代计算机网络中的 3 个极为重要的（并且在某种程度上是独立的）主题。在第 7 章中，我们研究了无线网络和移动网络，包括无线 LAN（其中有 WiFi 和蓝牙）、蜂窝电话网（包括 GSM、3G 和 4G）以及移动性（在 IP 网络和 GSM 网络中的）。在第 8 章中，我们首先学习加密和网络安全的基础知识，然后研究基础理论如何应用于各种各样的因特网环境。在最后一章（第 9 章）中，我们研究了音频和视频应用，例如因特网

电话、视频会议和流式存储媒体。此外，还讨论如何设计分组交换网络以对音频和视频应用程序提供一致的服务质量。

课后习题和问题

复习题

1.1节

R1. “主机”和“端系统”之间有什么不同？列举几种不同类型的端系统。Web服务器是一种端系统吗？

R2. “协议”一词常被用于描述外交关系。维基百科是怎样描述外交协议的？

R3. 标准对于协议为什么重要？

1.2节

R4. 列出6种接入技术。将它们分类为住宅接入、公司接入或广域无线接入。

R5. HFC传输速率在用户间是专用的还是共享的？在下行HFC信道中，可能出现碰撞吗？为什么？

R6. 列出你所在城市中的可供使用的住宅接入技术。对于每种类型的接入方式，给出所宣称的下行速率、上行速率和每月的价格。

R7. 以太LAN的传输速率是多少？

R8. 能够运行以太网的一些物理媒体是什么？

R9. 拨号调制解调器、HFC、DSL和FTTH都用于住宅接入。对于这些技术，给出每种技术的传输速率的范围，并讨论它们的传输速率是共享的还是专用的。

R10. 描述今天最为流行的无线因特网接入技术。对它们进行比较和对照。

1.3节

R11. 假定在发送主机和接收主机间只有一台分组交换机。发送主机和交换机间以及交换机和接收主机间的传输速率分别是 R_1 和 R_2。假设该交换机使用存储转发分组交换方式，发送一个长度为 L 的分组的端到端总时延是什么？（忽略排队时延、传播时延和处理时延。）

R12. 与分组交换网络相比，电路交换网络有哪些优点？在电路交换网络中，TDM比FDM有哪些优点？

R13. 假定用户共享一条2Mbps链路。同时假定当每个用户传输时连续以1Mbps传输，但每个用户仅传输20%的时间。

a. 当使用电路交换时，能够支持多少用户？

b. 作为该题的后继问题，假定使用分组交换。为什么如果两个或更少的用户同时传输的话，在链路前面基本上没有排队时延？为什么如果3个用户同时传输的话，将有排队时延？

c. 求出某指定用户正在传输的概率。

d. 假定现在有3个用户。求出在任何给定的时间，所有3个用户在同时传输的概率。求出队列增长的时间比率。

R14. 为什么等级结构中级别相同的两个ISP通常互相对等？某IXP是如何挣钱的？

R15. 某些内容提供商构建了自己的网络。描述谷歌的网络。内容提供商构建这些网络的动机是什么？

1.4节

R16. 考虑从某源主机跨越一条固定路由向某目的主机发送一分组。列出端到端时延中的时延组成成分。这些时延中的哪些是固定的，哪些是变化的？

R17. 访问配套Web网站上有关传输时延与传播时延的Java小程序。在速率、传播时延和可用的分组长度之中找出一种组合，使得该分组的第一个比特到达接收方之前发送方结束了传输。找出另一种组合，使得发送方完成传输之前，该分组的第一个比特到达了接收方。

R18. 一个长度为1000字节的分组经距离为2500km的链路传播，传播速率为 2.5×10^8 m/s并且传输速率为2Mbps，它需要用多长时间？更为一般地，一个长度为 L 的分组经距离为 d 的链路传播，传播速

率为 s 并且传输速率为 R bps，它需要用多长时间？该时延与传输速率相关吗？

R19. 假定主机 A 要向主机 B 发送一个大文件。从主机 A 到主机 B 的路径上有 3 段链路，其速率分别为 $R_1 = 500\text{kbps}$，$R_2 = 2\text{Mbps}$，$R_3 = 1\text{Mbps}$。

a. 假定该网络中没有其他流量，该文件传送的吞吐量是多少？

b. 假定该文件为 4MB。用吞吐量除以文件长度，将该文件传输到主机 B 大致需要多长时间？

c. 重复（a）和（b），只是这时 R_2 减小到 100kbps。

R20. 假定端系统 A 要向端系统 B 发送一个大文件。在一个非常高的层次上，描述端系统怎样从该文件生成分组。当这些分组之一到达某分组交换机时，该交换机使用分组中的什么信息来决定将该分组转发到哪一条链路上？因特网中的分组交换为什么可以与驱车从一个城市到另一个城市并沿途询问方向相类比？

R21. 访问配套 Web 站点的排队和丢包 Java 小程序。最大发送速率和最小的传输速率是多少？对于这些速率，流量强度是多大？用这些速率运行该 Java 小程序并确定出现丢包要花费多长时间？然后第二次重复该实验，再次确定出现丢包花费多长时间。这些值有什么不同？为什么会有这种现象？

1.5 节

R22. 列出一个层次能够执行的 5 个任务。这些任务中的一个（或两个）可能由两个（或更多）层次执行吗？

R23. 因特网协议栈中的 5 个层次有哪些？在这些层次中，每层的主要任务是什么？

R24. 什么是应用层报文？什么是运输层报文段？什么是网络层数据报？什么是链路层帧？

R25. 路由器处理因特网协议栈中的哪些层次？链路层交换机处理的是哪些层次？主机处理的是哪些层次？

1.6 节

R26. 病毒和蠕虫之间有什么不同？

R27. 描述如何产生一个僵尸网络，以及僵尸网络是怎样被用于 DDoS 攻击的。

R28. 假定 Alice 和 Bob 经计算机网络互相发送分组。假定 Trudy 将自己安置在网络中，使得她能够俘获由 Alice 发送的所有分组，并发送她希望给 Bob 的东西；她也能够俘获由 Bob 发送的所有分组，并发送她希望给 Alice 的东西。列出在这种情况下 Trudy 能够做的某些恶意的事情。

习题

P1. 设计并描述在自动柜员机和银行的中央计算机之间使用的一种应用层协议。你的协议应当允许验证用户卡和口令，查询账目结算（这些都在中央计算机中进行维护），支取账目（即向用户支付钱）。你的协议实体应当能够处理取钱时账目中钱不够的常见问题。通过列出自动柜员机和银行中央计算机在报文传输和接收过程中交换的报文和采取的动作来定义你的协议。使用类似于图 1-2 所示的图，拟定在简单无差错取钱情况下该协议的操作。明确地阐述在该协议中关于底层端到端运输服务所做的假设。

P2. 式（1-1）给出了经传输速率为 R 的 N 段链路发送长度 L 的一个分组的端到端时延。对于经过 N 段链路一个接一个地发送 P 个这样的分组，一般化地表示出这个公式。

P3. 考虑一个应用程序以稳定的速率传输数据（例如，发送方每 k 个时间单元产生一个 N 比特的数据单元，其中 k 较小且固定）。另外，当这个应用程序启动时，它将连续运行相当长的一段时间。回答下列问题，简要论证你的回答：

a. 是分组交换网还是电路交换网更为适合这种应用？为什么？

b. 假定使用了分组交换网，并且该网中的所有流量都来自如上所述的这种应用程序。此外，假定该应用程序数据传输速率的总和小于每条链路的各自容量。需要某种形式的拥塞控制吗？为什么？

P4. 考虑在图 1-13 中的电路交换网。回想在每条链路上有 4 条链路，以顺时针方向标记四台交换机 A、

B、C 和 D。

a. 在该网络中，任何时候能够进行同时连接的最大数量是多少？

b. 假定所有连接位于交换机 A 和 C 之间。能够进行同时连接的最大数量是多少？

c. 假定我们要在交换机 A 和 C 之间建立 4 条连接，在交换机 B 和 D 之间建立另外 4 条连接。我们能够让这些呼叫通过这 4 条链路建立路由以容纳所有 8 条连接吗？

P5. 回顾在 1.4 节中的车队的类比。假定传播速度为 100km/h。

a. 假定车队旅行 150km：在一个收费站前面开始，通过第二个收费站，并且正好在第三个收费站后面结束。其端到端时延是多少？

b. 重复（a），现在假定车队中有 8 辆汽车而不是 10 辆。

P6. 这个习题开始探讨传播时延和传输时延，这是数据网络中的两个重要概念。考虑两台主机 A 和 B 由一条速率为 R bps 的链路相连。假定这两台主机相隔 m 米，沿该链路的传播速率为 s m/s。主机 A 向主机 B 发送长度 L 比特的分组。

a. 用 m 和 s 来表示传播时延 d_{prop}。

b. 用 L 和 R 来确定该分组的传输时间 d_{trans}。

c. 忽略处理和排队时延，得出端到端时延的表达式。

d. 假定主机 A 在时刻 $t=0$ 开始传输该分组。在时刻 $t=d_{trans}$，该分组的最后一个比特在什么地方？

e. 假定 d_{prop} 大于 d_{trans}。在时刻 $t=d_{trans}$，该分组的第一个比特在何处？

f. 假定 d_{prop} 小于 d_{trans}。在时刻 $t=d_{trans}$，该分组的第一个比特在何处？

g. 假定 $s=2.5\times10^8$，$L=120$ 比特，$R=56$kbps。求出使 d_{prop} 等于 d_{trans} 的距离 m。

P7. 在这个习题中，我们考虑从主机 A 向主机 B 通过分组交换网发送语音（VoIP）。主机 A 将模拟语音转换为传输中的 64kbps 数字比特流。然后主机 A 将这些比特分为 56 字节的分组。A 和 B 之间有一条链路：它的传输速率是 2Mbps，传播时延是 10ms。一旦 A 收集了一个分组，就将它向主机 B 发送。一旦主机 B 接收到一个完整的分组，它将该分组的比特转换成模拟信号。从比特产生（从位于主机 A 的初始模拟信号起）的时刻起，到该比特被解码（在主机 B 上作为模拟信号的一部分），花了多少时间？

P8. 假定用户共享一条 3Mbps 的链路。又设每个用户传输时要求 150kbps，但是每个用户仅有 10% 的时间传输。（参见 1.3 节中关于“分组交换与电路交换的对比”的讨论。）

a. 当使用电路交换时，能够支持多少用户？

b. 对于本习题的后续小题，假定使用分组交换。求出某给定用户正在传输的概率。

c. 假定有 120 个用户。求出在任何给定时刻，实际有 n 个用户在同时传输的概率。（提示：使用二项式分布。）

d. 求出有 21 个或更多用户同时传输的概率。

P9. 考虑在 1.3 节“分组交换与电路交换的对比”的讨论中，给出了一个具有一条 1Mbps 链路的例子。用户在忙时以 100kbps 速率产生数据，但忙时仅以 $p=0.1$ 的概率产生数据。假定用 1Gbps 链路替代 1Mbps 的链路。

a. 当采用电路交换技术时，能被同时支持的最大用户数量 N 是多少？

b. 现在考虑分组交换和有 M 个用户的情况。给出多于 N 用户发送数据的概率公式（用 p、M、N 表示）。

P10. 考虑一个长度为 L 的分组从端系统 A 开始，经 3 段链路传送到目的端系统。令 d_i、s_i 和 R_i 表示链路 i 的长度、传播速度和传输速率（$i=1, 2, 3$）。该分组交换机对每个分组的时延为 d_{proc}。假定没有排队时延，用 d_i、s_i、R_i（$i=1, 2, 3$）和 L 表示，该分组总的端到端时延是什么？现在假定该分组是 1500 字节，在所有 3 条链路上的传播时延是 2.5×10^8m/s，所有 3 条链路的传输速率是 2Mbps，分组交换机的处理时延是 3ms，第一段链路的长度是 5000km，第二段链路的长度是 4000km，并且最后一段链路的长度是 1000km。对于这些值，该端到端时延为多少？

P11. 在上述习题中，假定 $R_1=R_2=R_3=R$ 且 $d_{proc}=0$。进一步假定该分组交换机不存储转发分组，而是

在等待分组到达前立即传输它收到的每个比特。这时端到端时延为多少？

P12. 一台分组交换机接收一个分组并决定该分组应当转发的出链路。当某分组到达时，另一个分组正在该出链路上被发送到一半，还有 4 个其他分组正等待传输。这些分组以到达的次序传输。假定所有分组是 1500 字节并且链路速率是 2Mbps。该分组的排队时延是多少？在更一般的情况下，当所有分组的长度是 L，传输速率是 R，当前正在传输的分组已经传输了 x 比特，并且已经在队列中有 n 个分组，其排队时延是多少？

P13. a. 假定有 N 个分组同时到达一条当前没有分组传输或排队的链路。每个分组长为 L，链路传输速率为 R。对 N 个分组而言，其平均排队时延是多少？

b. 现在假定每隔 LN/R 秒有 N 个分组同时到达链路。一个分组的平均排队时延是多少？

P14. 考虑某路由器缓存中的排队时延。令 I 表示流量强度；即 $I = La/R$。假定排队时延的形式为 $IL/R(1-I)$，其中 $I<1$。

a. 写出总时延即排队时延加上传输时延的公式。

b. 以 L/R 为函数画出总时延的图。

P15. 令 a 表示在一条链路上分组的到达率（以分组/秒计），令 μ 表示一条链路上分组的传输率（以分组/秒计）。基于上述习题中推导出的总时延公式（即排队时延加传输时延），推导出以 a 和 μ 表示的总时延公式。

P16. 考虑一台路由器缓存前面的一条出链路。在这个习题中，将使用李特尔（Little）公式，这是排队论中的一个著名公式。令 N 表示在缓存中的分组加上被传输的分组的平均数。令 a 表示到达该链路的分组速率。令 d 表示一个分组历经的平均总时延（即排队时延加传输时延）。李特尔公式是 $N = a \times d$。假定该缓存平均包含 10 个分组，并且平均分组排队时延是 10ms。该链路的传输速率是 100 分组/秒。使用李特尔公式，在没有丢包的情况下，平均分组到达率是多少？

P17. a. 对于不同的处理速率、传输速率和传播时延，给出 1.4.3 节中式（1-2）的一般表达式。

b. 重复（a），不过此时假定在每个节点有平均排队时延 d_{queue}。

P18. 在一天的 3 个不同的小时内，在同一个大陆上的源和目的地之间执行 Traceroute。

a. 在这 3 个小时的每个小时中，求出往返时延的均值和方差。

b. 在这 3 个小时的每个小时中，求出路径上的路由器数量。在这些时段中，该路径发生变化了吗？

c. 试图根据源到目的地 Traceroute 分组通过的情况，辨明 ISP 网络的数量。具有类似名字和/或类似的 IP 地址的路由器应当被认为是同一个 ISP 的一部分。在你的实验中，在相邻的 ISP 间的对等接口处出现最大的时延了吗？

d. 对位于不同大陆上的源和目的地重复上述内容。比较大陆内部和大陆之间的这些结果。

P19. a. 访问站点 www.traceroute.org，并从法国两个不同的城市向位于美国的相同的目的主机执行 Traceroute。在这两个 Traceroute 中，有多少条链路是相同的？大西洋沿岸国家的链路相同吗？

b. 重复（a），但此时选择位于法国的一个城市和位于德国的另一个城市。

c. 在美国挑选一个城市，然后向位于中国的两个不同城市的主机执行 Traceroute。在这两次 Traceroute 中有多少链路是相同的？在到达中国前这两个 Traceroute 分开了吗？

P20. 考虑对应于图 1-20b 吞吐量的例子。现在假定有 M 对客户－服务器而不是 10 对。用 R_s、R_c 和 R 分别表示服务器链路、客户链路和网络链路的速率。假设所有的其他链路都有充足容量，并且除了由这 M 对客户－服务器产生的流量外，网络中没有其他流量。推导出由 R_s、R_c、R 和 M 表示的通用吞吐量表达式。

P21. 考虑图 1-19b。现在假定在服务器和客户之间有 M 条路径。任两条路径都不共享任何链路。路径 $k(k=1, \cdots, M)$ 由传输速率为 R_1^k，R_2^k，$\cdots$，R_N^k 的 N 条链路组成。如果服务器仅能够使用一条路径向客户发送数据，则该服务器能够取得的最大吞吐量是多少？如果该服务器能够使用所有 M 条路径发送数据，则该服务器能够取得的最大吞吐量是多少？

P22. 考虑图 1-19b。假定服务器与客户之间的每条链路的丢包概率为 p，且这些链路的丢包率是独立的。一个（由服务器发送的）分组成功地被接收方收到的概率是多少？如果在从服务器到客户的路径上

分组丢失了，则服务器将重传该分组。平均来说，为了使客户成功地接收该分组，服务器将要重传该分组多少次？

P23. 考虑图1-19a。假定我们知道沿着从服务器到客户的路径的瓶颈链路是速率为 R_s bps 的第一段链路。假定我们从服务器向客户发送紧密相连的一对分组，且沿这条路径没有其他流量。假定每个分组的长度为 L 比特，两条链路具有相同的传播时延 d_{prop}。

a. 在目的地，分组的到达间隔时间有多大？也就是说，从第一个分组的最后一个比特到达到第二个分组最后一个比特到达所经过的时间有多长？

b. 现在假定第二段链路是瓶颈链路（即 $R_c < R_s$）。第二个分组在第二段链路输入队列中排队是可能的吗？请解释原因。现在假定服务器在发送第一个分组 T 秒之后再发送第二个分组。为确保在第二段链路之前没有排队，T 必须要有多长？试解释原因。

P24. 假设你希望从波士顿向洛杉矶紧急传送 40×10^{12} 字节数据。你有一条100Mbps专用链路可用于传输数据。你是愿意通过这条链路传输数据，还是愿意使用 FedEx 夜间快递来交付？解释你的理由。

P25. 假定两台主机A和B相隔20 000km，由一条直接的 $R = 2$Mbps 的链路相连。假定跨越该链路的传播速率是 2.5×10^8m/s。

a. 计算带宽-时延积 $R \cdot t_{prop}$。

b. 考虑从主机A到主机B发送一个800 000比特的文件。假定该文件作为一个大的报文连续发送。在任何给定的时间，在链路上具有的比特数量最大值是多少？

c. 给出带宽-时延积的一种解释。

d. 在该链路上一个比特的宽度（以米计）是多少？它比一个足球场更长吗？

e. 用传播速率 s、带宽 R 和链路 m 的长度表示，推导出一个比特宽度的一般表示式。

P26. 对于习题P25，假定我们能够修改 R。对什么样的 R 值，一个比特的宽度能与该链路的长度一样长？

P27. 考虑习题P25，但此时链路的速率是 $R = 1$Gbps。

a. 计算带宽-时延积 $R \cdot d_{prop}$。

b. 考虑从主机A到主机B发送一个800 000比特的文件。假定该文件作为一个大的报文连续发送。在任何给定的时间，在链路上具有的比特数量最大值是多少？

c. 在该链路上一个比特的宽度（以米计）是多少？

P28. 再次考虑习题P25。

a. 假定连续发送，发送该文件需要多长时间？

b. 假定现在该文件被划分为20个分组，每个分组包含40 000比特。假定每个分组被接收方确认，确认分组的传输时间可忽略不计。最后，假定前一个分组被确认后，发送方才能发送分组。发送该文件需要多长时间？

c. 比较（a）和（b）的结果。

P29. 假定在同步卫星和它的地球基站之间有一条10Mbps的微波链路。每分钟该卫星拍摄一幅数字照片，并将它发送到基站。假定传播速率是 2.4×10^8m/s。

a. 该链路的传播时延是多少？

b. 带宽-时延积 $R \cdot d_{prop}$ 是多少？

c. 若 x 表示该照片的大小。对于这条微波链路，能够连续传输的 x 最小值是多少？

P30. 考虑1.5节中我们在分层讨论中对航空旅行的类比，随着协议数据单元向协议栈底层流动，首部在增加。随着旅客和行李移动到航线协议栈底部，有与上述首部信息等价的概念吗？

P31. 在包括因特网的现代分组交换网中，源主机将长应用层报文（如一个图像或音乐文件）分段为较小的分组并向网络发送。接收方则将这些分组重新装配为初始报文。我们称这个过程为**报文分段**。图1-27显示了一个报文在报文不分段或报文分段情况下的端到端传输。考虑一个长度为 8×10^6 比特的报文，它在图1-27中从源发送到目的地。假定在该图中的每段链路是2Mbps。忽略传播、排队和处理时延。

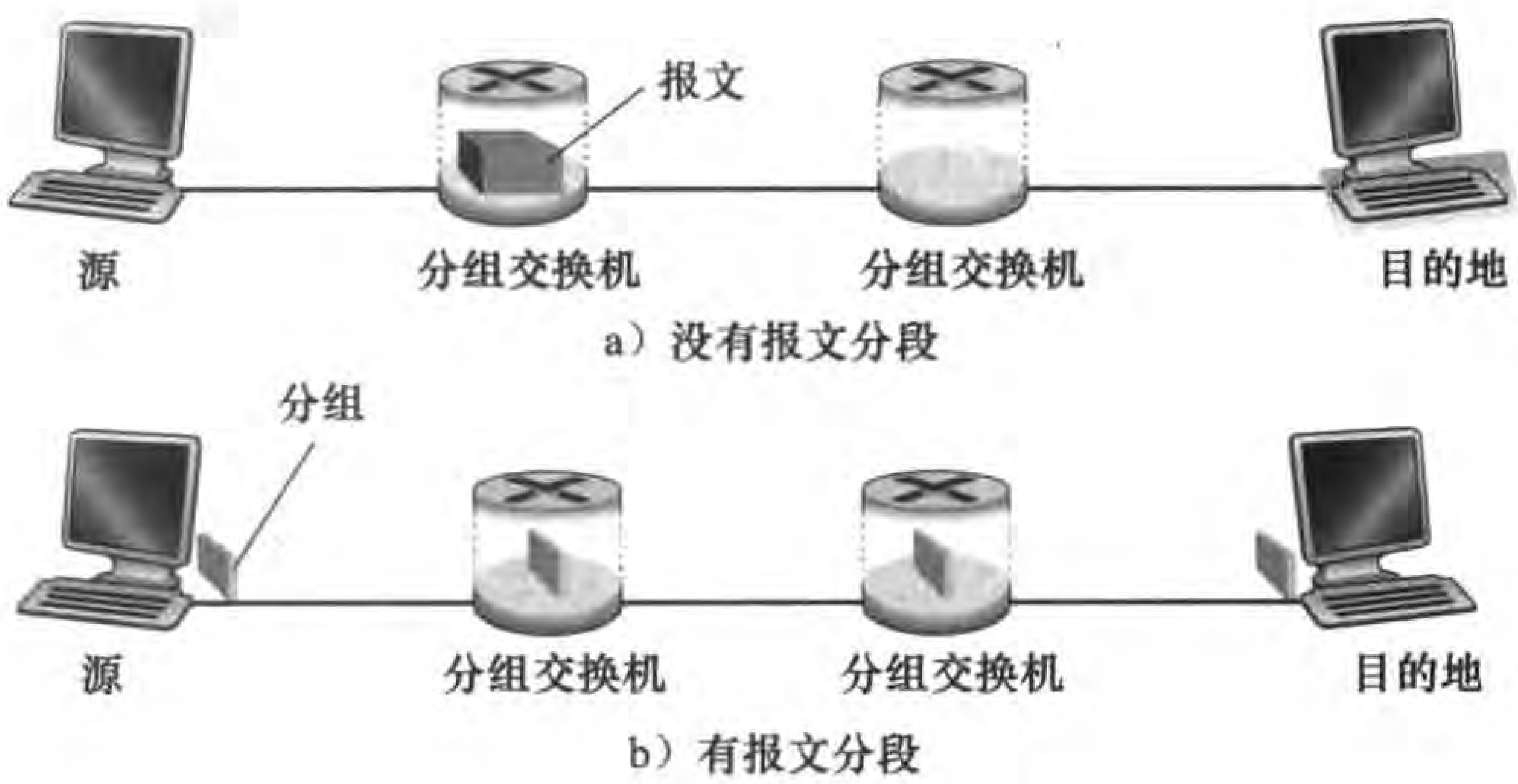


图 1-27 端到端报文传输

a. 考虑从源到目的地发送该报文且没有报文分段。从源主机到第一台分组交换机移动报文需要多长时间？记住，每台交换机均使用存储转发分组交换，从源主机移动该报文到目的主机需要多长时间？

b. 现在假定该报文被分段为 800 个分组，每个分组 10 000 比特长。从源主机移动第一个分组到第一台交换机需要多长时间？从第一台交换机发送第一个分组到第二台交换机，从源主机发送第二个分组到第一台交换机各需要多长时间？什么时候第二个分组能被第一台交换机全部收到？

c. 当进行报文分段时，从源主机向目的主机移动该文件需要多长时间？将该结果与（a）的答案进行比较并解释之。

d. 除了减小时延外，使用报文分段还有什么原因？

e. 讨论报文分段的缺点。

P32. 用本书的 Web 网站上的报文分段小 Java 小程序进行实验。该 Java 程序中的时延与前一个习题中的时延相当吗？链路传播时延是怎样影响分组交换（有报文分段）和报文交换的端到端总时延的？

P33. 考虑从主机 A 到主机 B 发送一个 F 比特的大文件。A 和 B 之间有三段链路（和两台交换机），并且该链路不拥塞（即没有排队时延）。主机 A 将该文件分为每个为 S 比特的报文段，并为每个报文段增加一个 80 比特的首部，形成 $L=80+S$ 比特的分组。每条链路的传输速率为 R bps。求出从 A 到 B 移动该文件时延最小的值 S。忽略传播时延。

P34. Skype 提供了一种服务，使你能用 PC 向普通电话打电话。这意味着语音呼叫必须通过因特网和电话网。讨论这是如何做到的。

Wireshark 实验

“不闻不若闻之，闻之不若见之，见之不若知之，知之不若行之。”

——中国谚语

一个人通常能够通过以下方法加深对网络协议的理解：观察它们的动作和经常摆弄它们，即观察两个协议实体之间交换的报文序列，钻研协议运行的细节，使协议执行某些动作，观察这些动作及其后果。这能够在仿真环境下或在如因特网这样的真实网络环境下完成。在本书配套 Web 站点上的 Java 小程序采用的是第一种方法。在 Wireshark 实验中，我们将采用后一种方法。你可以在家中或实验室中使用桌面计算机在各种情况下运行网络应用程序。在你的计算机上观察网络协议，它是如何与在因特网别处执行的协议实体交互和交换报文的。因此，你与你的计算机将是这些真实实验的有机组成部分。你将通过动手来观察和学习。

用来观察执行协议实体之间交换的报文的基本工具称为**分组嗅探器**（packet sniffer）。顾名思义，一个分组嗅探器被动地拷贝（嗅探）由你的计算机发送和接收的报文；它也能显示出这些被捕获报文的各个协议字段的内容。图 1-28 中显示了 Wireshark 分组嗅探器的屏幕快照。Wireshark 是一个运行在 Win-

dows、Linux/Unix 和 Mac 计算机上的免费分组嗅探器。贯穿全书，你将发现 Wireshark 实验能让你探索在该章中学习的一些协议。在这第一个 Wireshark 实验中，你将获得并安装一个 Wireshark 的副本，访问一个 Web 站点，捕获并检查在你的 Web 浏览器和 Web 服务器之间交换的协议报文。

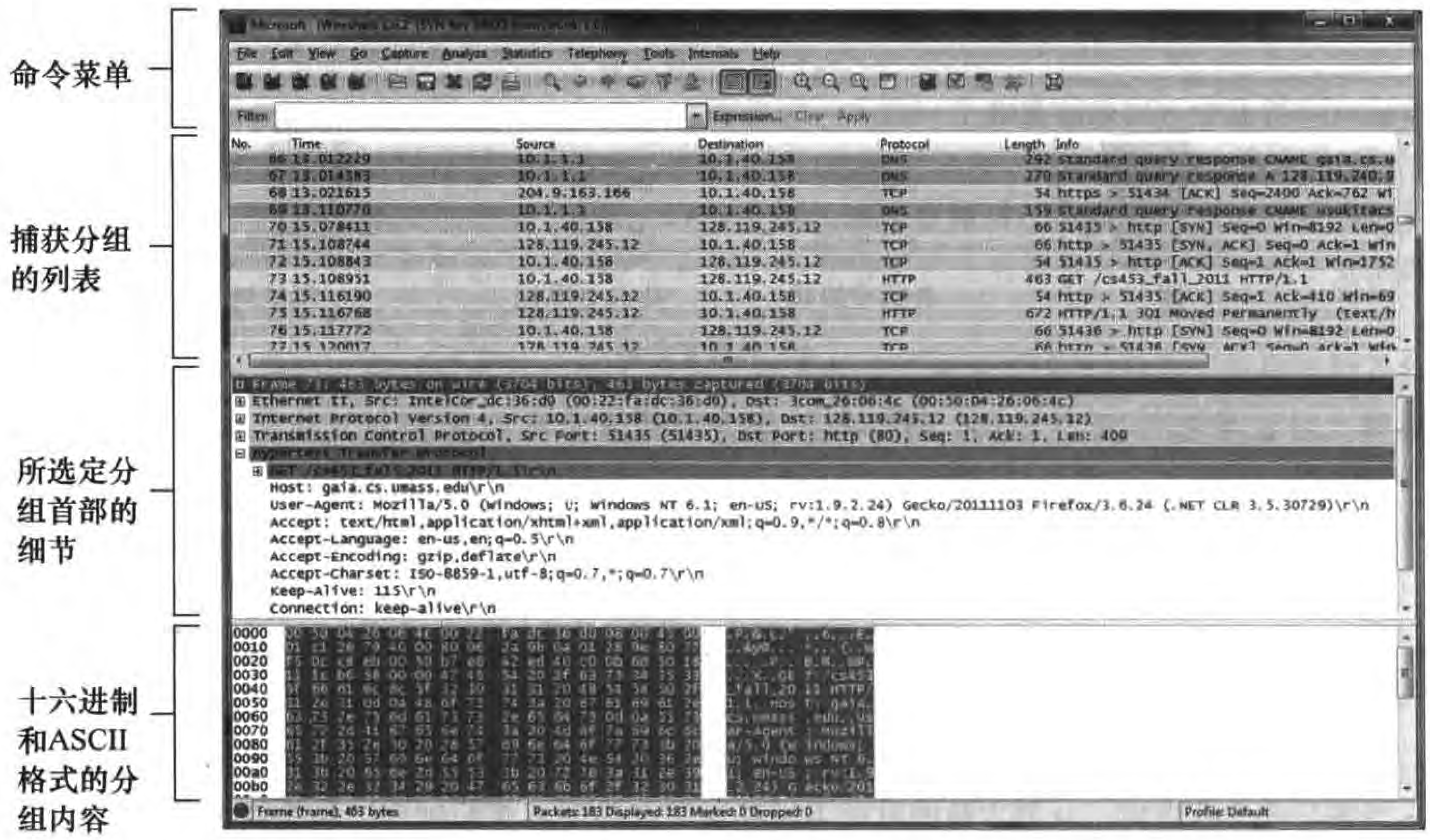


图 1-28　一个 Wireshark 屏幕快照（打印的 Wireshark 屏幕快照得到了 Wireshark 基金会的许可）

你能够在 Web 站点 http://www.pearsonhighered.com/cs-resources/上找到有关该第一个 Wireshark 实验的全部材料（包括如何获得并安装 Wireshark 的指导）。

人物专访

Leonard Kleinrock 是加州大学洛杉矶分校（UCLA）的计算机科学教授。1969 年，他在 UCLA 的计算机成为因特网的第一个节点。1961 年，他创造的分组交换原理成为因特网的支撑技术。他在纽约城市大学（City College of New York，CCNY）获得电气工程学士学位，并在麻省理工学院（MIT）获得电气工程硕士和博士学位。



Leonard Kleinrock

- 是什么使得您决定专门研究网络/因特网技术的?

当我于 1959 年在 MIT 读博士时，我发现周围的大多数同学正在信息理论和编码理论领域做研究。在 MIT，那时有伟大的研究者 Claude Shannon，他已经开创这些领域，并且已经解决了许多重要的问题。留下来的研究问题既难又不太重要。因此我决定开始新的研究领域，而该领域其他人还没有想到。回想那时在 MIT 我的周围有许多计算机，我很清楚很快这些计算机将有相互通信的需求。在那时，却没有有效的办法来做到这一点，因此我决定研发能够创建有效的数据网络的技术。

- 您在计算机产业的第一份工作是什么？它使您有哪些收益?

1951～1957 年，我为了获得电子工程学士学位在 CCNY 读夜大。在那段时间里，我在一家称为 Photobell 的工业电子小公司工作，先是当技术员，然后当工程师。在那里，我在它们的产品线上引入了数字技术。我们主要使用光电子设备来检测某些物体（盒子、人等）的存在，一种称为*双稳态多频振荡器*的电路的使用正是我们需要的技术类型，它能将数字处理引入检测领域。这些电路恰好是计算机的基本模块，用今天的话说就是*触发电路或交换器*。

- 当您发送第一个主机到主机报文（从 UCLA 到斯坦福研究院）时，您心中想到了什么?

坦率地说，我们当时并没有想到那件事的重要性。我们没有准备具有历史意义的豪言壮语，就像昔日许多发明家所做的那样（如塞缪尔·莫尔斯的“上帝创造了什么（What hath God wrought）”，亚历山大·格瑞汉姆·贝尔的“Watson先生，请来这里！我想见你”，或尼尔·阿姆斯特朗的“个人的一小步，人类的一大步”）。多么聪明的人哪！他们明白媒体和公众的关系。我们要做的所有工作是向斯坦福研究院的计算机进行注册。当我们键入“L”，它被正确收到，当我们键入“o”又被正确收到，而当我们键入“g”则引起斯坦福研究院主机的崩溃！因此，这将我们的报文转换为最短的，也许是最有预测性的报文，“Lo!”即为“真想不到（Lo and behold）!”。

那年早些时候，UCLA新闻稿引用我的话说，一旦该网络建立并运行起来，将可能从我们的家中和办公室访问计算机设施，就像我们获得电力和电话连接那样容易。因此那时我的美好愿望是，因特网将是一个无所不在的、总是运行的、总是可用的网络，任何人从任何地方用任何设备将能够与之相连，并且它将是不可见的。然而，我从没有期待我的99岁的母亲将能够上因特网，但她的确做到了这一点。

- 您对未来网络的展望是什么？

我的展望中最容易的部分是预测基础设施本身。我预见我们看到移动计算、移动设备和智能空间的大量部署。轻量级、廉价、高性能、便携的计算和通信设备（加上因特网的无处不在）的确使我们成为游牧一员。游牧计算是指使从一个地方旅行到另一个地方的端用户，以透明方式访问因特网服务，无论他们旅行到何处，无论他们携带什么设备或获得何种接入。展望中最困难的部分是预测应用和服务，它们以引人注目的方式不断地带给我们惊喜（电子邮件、搜索技术、万维网、博客、社交网络、用户一代以及音乐、照片和视频等的共享）。我们正面临一种新的惊奇和创新，即移动应用装载于手持设备中。

下一步将使我们从信息空间虚拟世界（netherworld）移动到智能空间的物理世界。我们的环境（办公桌、墙壁、车辆、钟表、腰带等）将因技术而栩栩如生，这些技术包括执行器（actuator）、传感器、逻辑、处理、存储、照相机、麦克风、话筒、显示器和通信。这种嵌入式技术将使得环境能提供我们需要的IP服务。当我走进一间房间时，该房间知道我的到来。我将能够与环境自然地通信，如同说英语一样；我的请求产生的响应将从墙上的显示器通过我的眼镜以Web网页的形式呈现给我，就像说话、全息照相等一样。

再向前看一点，我看到未来的网络包括了下列附加的关键组件。我看到在网络各处部署的智能软件代理，它的功能是挖掘数据，根据数据采取动作，观察趋势，并能动态地、自适应地执行任务。我看到相当多的网络流量并不是由人产生的，而是由这些嵌入式设备和这些智能软件代理产生的。我看到大批的自组织系统控制这个巨大、快速的网络。我看到巨量的信息瞬间通过网络立即得到强力处理和过滤。因特网最终将是一个无所不在的全球性神经系统。当我们急速在21世纪进发时，我将看到这些东西和更多的东西。

- 是谁激发了您的职业灵感？

到目前为止，是麻省理工学院的Claude Shannon。他是一名卓越的研究者，具有以高度直觉的方式将他的数学理念与物理世界关联起来的能力。他是我的博士论文答辩委员会的成员。

- 您对进入网络/因特网领域的学生有什么忠告吗？

因特网和由它使能的所有东西是一个巨大的新前沿，充满了令人惊奇的挑战，为众多创新提供了广阔空间。不要受今天技术的束缚，开动大脑，想象能够做些什么，并去实现它。

第2章

Computer Networking: A Top-Down Approach, Seventh Edition

应 用 层

网络应用是计算机网络存在的理由，如果我们不能构想出任何有用的应用，也就没有任何必要去设计支持它们的网络协议了。自因特网全面发展以来，的确已开发出众多有用的、有趣的网络应用。这些应用程序已经成为因特网成功的驱动力，激励人们在家庭、学校、政府和商业中利用网络，使因特网成为他们日常活动的密不可分的一部分。

因特网应用包括：20 世纪 70 年代和 80 年代开始流行的经典的基于文本的应用，如文本电子邮件、远程访问计算机、文件传输和新闻组；20 世纪 90 年代中期引入的招人喜爱的应用——万维网，包括 Web 冲浪、搜索和电子商务；20 世纪末引入的两个招人喜爱的应用——即时讯息和对等（P2P）文件共享。自 2000 年以来，新型和极其引人入胜的应用持续出现，包括：IP 电话（VoIP）、视频会议（如 Skype、Facetime 和 Google Hangouts）；用户生成的视频（如 YouTube）和点播电影（如 Netflix）；多方在线游戏（如《第二人生》（Second Life）和《魔兽世界》（World of Warcraft））。在这段时期，我们看到了新一代社交网络应用，如 Facebook、Instagram、Twitter 和微信，它们在因特网的网络或路由器和通信链路之上创建了引人入胜的人类网络。近年来，随着智能手机的发展，出现了大量基于位置的移动应用程序，包括流行的签到、约会和道路流量预测应用（如 Yelp、Tinder、Waz 和 Yik Yak）。显然，新型和令人兴奋的因特网应用的步伐并没有减缓。也许本书的一些读者将会创建下一代招人喜爱的因特网应用。

在本章中，我们学习有关网络应用的原理和实现方面的知识。我们从定义关键的应用层概念开始，其中包括应用程序所需要的网络服务、客户和服务器、进程和运输层接口。我们详细考察几种网络应用程序，包括 Web、电子邮件、DNS、对等文件分发和视频流（第 9 章关注多媒体应用，包括流式视频和 VoIP）。然后我们将涉及开发运行在 TCP 和 UDP 上的网络应用程序。特别是，我们学习套接字接口，并浮光掠影地学习用 Python 语言写的一些简单的客户 - 服务器应用程序。在本章结尾，我们也将提供几个有趣、有意思的套接字编程作业。

应用层是我们学习协议非常好的起点，它最为我们所熟悉。我们熟悉的很多应用就是建立在这些将要学习的协议基础上的。通过对应用层的学习，将有助于我们很好地感受到协议的方方面面，将使我们了解到很多问题，这些问题在我们学习运输层、网络层及数据链路层协议时也同样会碰到。

2.1 应用层协议原理

假定你对新型网络应用有了一些想法。也许这种应用将为人类提供一种伟大的服务，或者将使你的教授高兴，或者将带给你大量的财富，或者只是在开发中获得乐趣。无论你的动机是什么，我们现在考察一下如何将你的想法转变为一种真实世界的网络应用。

研发网络应用程序的核心是写出能够运行在不同的端系统和通过网络彼此通信的程序。例如，在 Web 应用程序中，有两个互相通信的不同的程序：一个是运行在用户主机（桌面

机、膝上机、平板电脑、智能电话等）上的浏览器程序；另一个是运行在 Web 服务器主机上的 Web 服务器程序。另一个例子是 P2P 文件共享系统，在参与文件共享的社区中的每台主机中都有一个程序。在这种情况下，在各台主机中的这些程序可能都是类似的或相同的。

因此，当研发新应用程序时，你需要编写将在多台端系统上运行的软件。该软件能够用如 C、Java 或 Python 来编写。重要的是，你不需要写在网络核心设备如路由器或链路层交换机上运行的软件。即使你要为网络核心设备写应用程序软件，你也不能做到这一点。如我们在第 1 章所知，以及如图 1-24 所显示的那样，网络核心设备并不在应用层上起作用，而仅在较低层起作用，特别是在网络层及下面层次起作用。这种基本设计，即将应用软件限制在端系统（如图 2-1 所示）的方法，促进了大量的网络应用程序的迅速研发和部署。

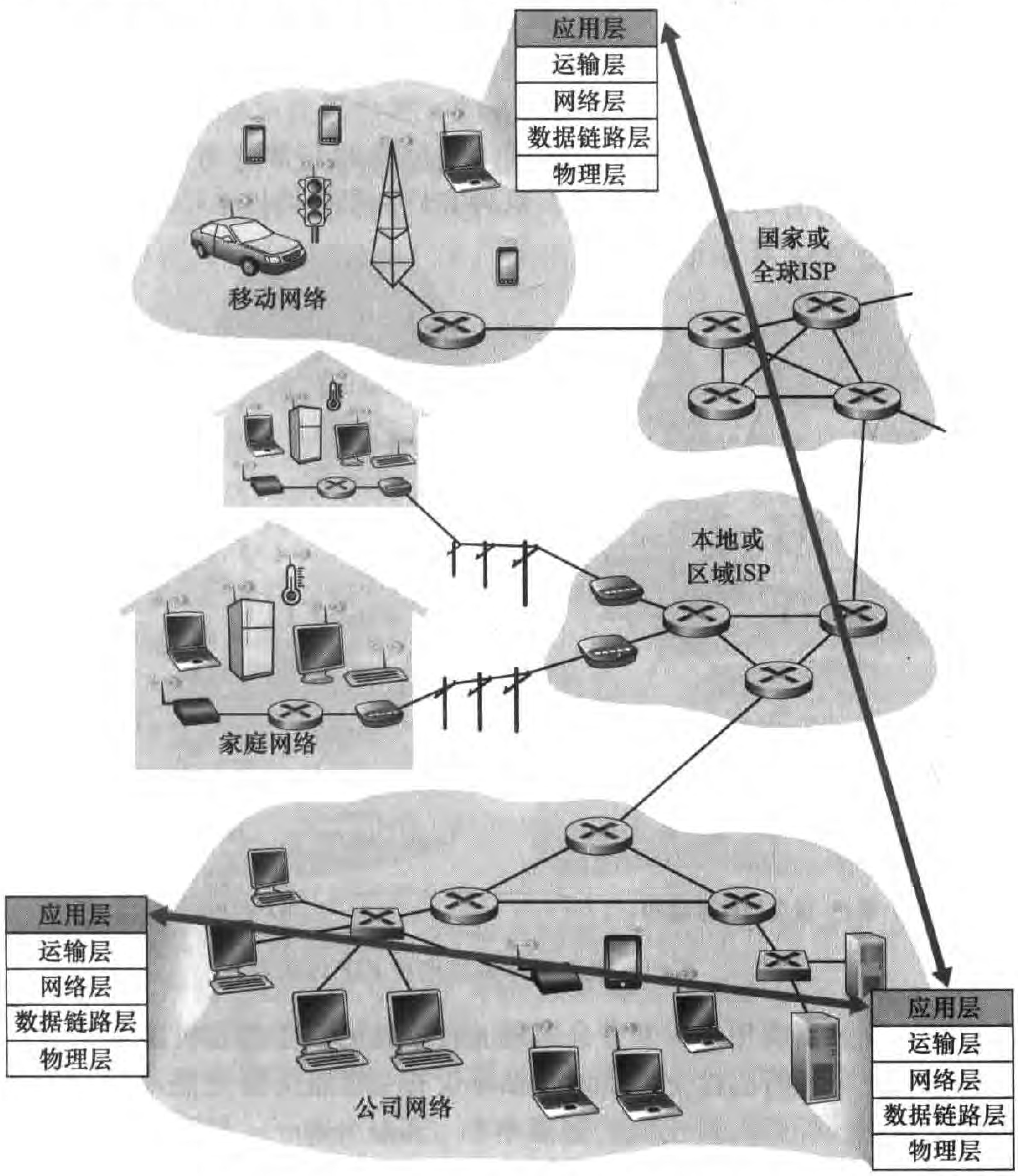


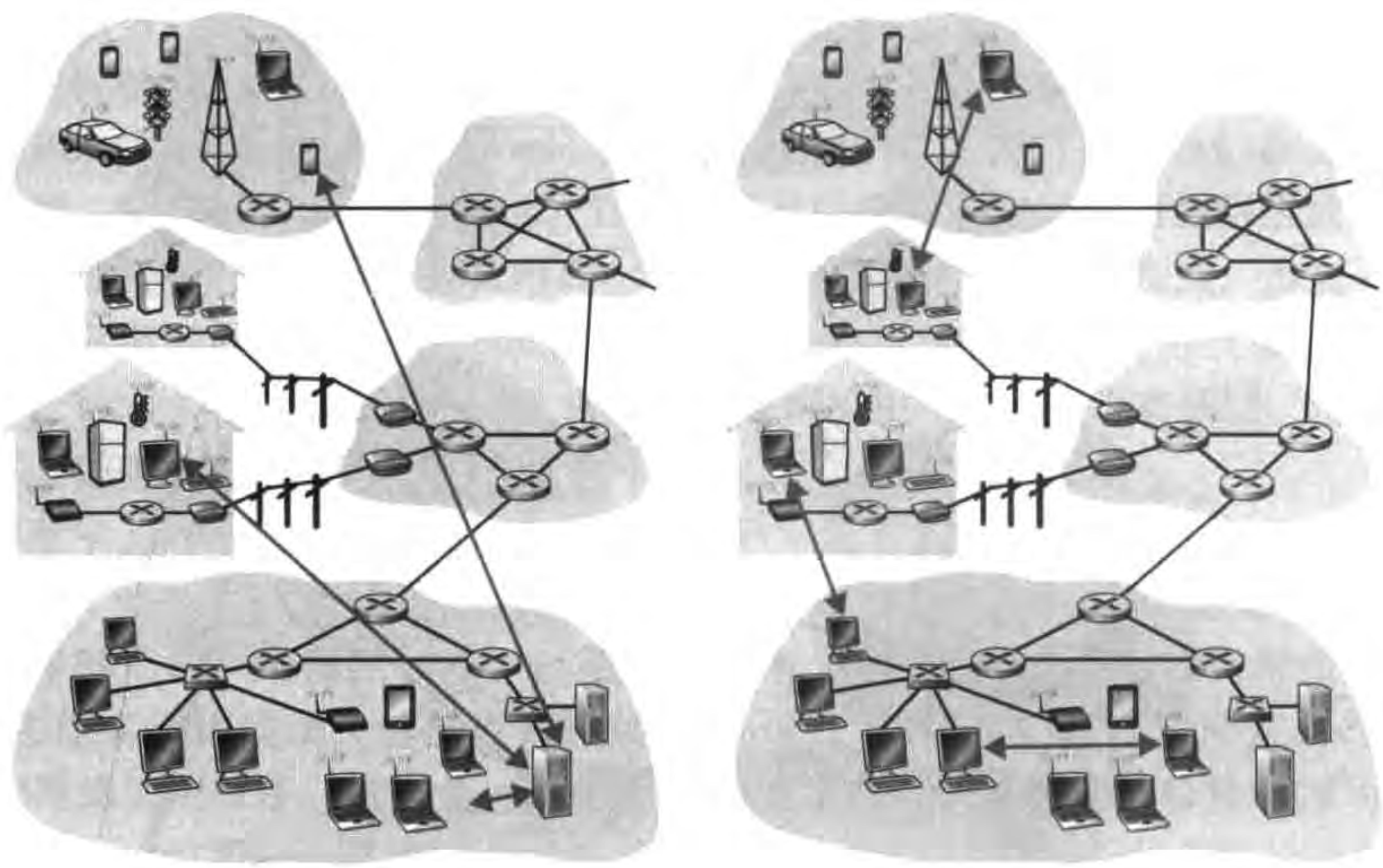
图 2-1 在应用层的端系统之间的网络应用的通信

2.1.1 网络应用程序体系结构

当进行软件编码之前，应当对应用程序有一个宽泛的体系结构计划。记住应用程序的体系结构明显不同于网络的体系结构（例如在第 1 章中所讨论的 5 层因特网体系结构）。从应用程序研发者的角度看，网络体系结构是固定的，并为应用程序提供了特定的服务集

合。在另一方面，**应用程序体系结构**（application architecture）由应用程序研发者设计，规定了如何在各种端系统上组织该应用程序。在选择应用程序体系结构时，应用程序研发者很可能利用现代网络应用程序中所使用的两种主流体系结构之一：客户-服务器体系结构或对等（P2P）体系结构。

在客户-服务器体系结构（client-server architecture）中，有一个总是打开的主机称为**服务器**，它服务于来自许多其他称为客户的主机的请求。一个典型的例子是Web应用程序，其中总是打开的Web服务器服务于来自浏览器（运行在客户主机上）的请求。当Web服务器接收到来自某客户对某对象的请求时，它向该客户发送所请求的对象作为响应。值得注意的是利用客户-服务器体系结构，客户相互之间不直接通信；例如，在Web应用中两个浏览器并不直接通信。客户-服务器体系结构的另一个特征是该服务器具有固定的、周知的地址，该地址称为IP地址（我们将很快讨论它）。因为该服务器具有固定的、周知的地址，并且因为该服务器总是打开的，客户总是能够通过向该服务器的IP地址发送分组来与其联系。具有客户-服务器体系结构的非常著名的应用程序包括Web、FTP、Telnet和电子邮件。图2-2a中显示了这种客户-服务器体系结构。



a）客户-服务器体系结构　　b）P2P体系结构

图2-2　客户-服务器体系结构及P2P体系结构

在一个客户-服务器应用中，常常会出现一台单独的服务器主机跟不上它所有客户请求的情况。例如，一个流行的社交网络站点如果仅有一台服务器来处理所有请求，将很快变得不堪重负。为此，配备大量主机的**数据中心**（data center）常被用于创建强大的虚拟服务器。最为流行的因特网服务——如搜索引擎（如谷歌、Bing和百度）、因特网商务（如亚马逊、e-Bay和阿里巴巴）、基于Web的电子邮件（如Gmail和雅虎邮件）、社交网络（如脸书、Instagram、推特和微信），就应用了一个或多个数据中心。如在1.3.3节中所讨论的那样，谷歌有分布于全世界的30～50个数据中心，这些数据中心共同处理搜索、YouTube、Gmail和其他服务。一个数据中心能够有数十万台服务器，它们必须要供电和维护。此外，服务提供商必须支付不断出现的互联和带宽费用，以发送和接收到达/来自数据中心的数据。

在一个 **P2P 体系结构**（P2P architecture）中，对位于数据中心的专用服务器有最小的（或者没有）依赖。相反，应用程序在间断连接的主机对之间使用直接通信，这些主机对被称为对等方。这些对等方并不为服务提供商所有，相反却为用户控制的桌面机和膝上机所有，大多数对等方驻留在家庭、大学和办公室。因为这种对等方通信不必通过专门的服务器，该体系结构被称为对等方到对等方的。许多目前流行的、流量密集型应用都是 P2P 体系结构的。这些应用包括文件共享（例如 BitTorrent）、对等方协助下载加速器（例如迅雷）、因特网电话和视频会议（例如 Skype）。图 2-2b 中显示了 P2P 的体系结构。需要提及的是，某些应用具有混合的体系结构，它结合了客户 - 服务器和 P2P 的元素。例如，对于许多即时讯息应用而言，服务器被用于跟踪用户的 IP 地址，但用户到用户的报文在用户主机之间（无须通过中间服务器）直接发送。

P2P 体系结构的最引人入胜的特性之一是它们的**自扩展性**（self-scalability）。例如，在一个 P2P 文件共享应用中，尽管每个对等方都由于请求文件产生工作负载，但每个对等方通过向其他对等方分发文件也为系统增加服务能力。P2P 体系结构也是有成本效率的，因为它们通常不需要庞大的服务器基础设施和服务器带宽（这与具有数据中心的客户 - 服务器设计形成鲜明对比）。然而，未来 P2P 应用由于高度非集中式结构，面临安全性、性能和可靠性等挑战。

2.1.2 进程通信

在构建网络应用程序前，还需要对运行在多个端系统上的程序是如何互相通信的情况有一个基本了解。用操作系统的术语来说，进行通信的实际上是**进程**（process）而不是程序。一个进程可以被认为是运行在端系统中的一个程序。当多个进程运行在相同的端系统上时，它们使用进程间通信机制相互通信。进程间通信的规则由端系统上的操作系统确定。而在本书中，我们并不特别关注同一台主机上的进程间的通信，而关注运行在不同端系统（可能具有不同的操作系统）上的进程间的通信。

在两个不同端系统上的进程，通过跨越计算机网络交换**报文**（message）而相互通信。发送进程生成并向网络中发送报文；接收进程接收这些报文并可能通过回送报文进行响应。图 2-1 显示了驻留在 5 层协议栈的应用层进程互相通信的情况。

1. 客户和服务器进程

网络应用程序由成对的进程组成，这些进程通过网络相互发送报文。例如，在 Web 应用程序中，一个客户浏览器进程与一台 Web 服务器进程交换报文。在一个 P2P 文件共享系统中，文件从一个对等方中的进程传输到另一个对等方中的进程。对每对通信进程，我们通常将这两个进程之一标识为**客户**（client），而另一个进程标识为**服务器**（server）。对于 Web 而言，浏览器是一个客户进程，Web 服务器是一台服务器进程。对于 P2P 文件共享，下载文件的对等方标识为客户，上载文件的对等方标识为服务器。

你或许已经观察到，如在 P2P 文件共享的某些应用中，一个进程能够既是客户又是服务器。在 P2P 文件共享系统中，一个进程的确既能上载文件又能下载文件。无论如何，在任何给定的一对进程之间的通信会话场景中，我们仍能将一个进程标识为客户，另一个进程标识为服务器。我们定义客户和服务器进程如下：

> 在一对进程之间的通信会话场景中，发起通信（即在该会话开始时发起与其他进程的联系）的进程被标识为**客户**，在会话开始时等待联系的进程是**服务器**。

在 Web 中，一个浏览器进程向一台 Web 服务器进程发起联系，因此该浏览器进程是客户，而该 Web 服务器进程是服务器。在 P2P 文件共享中，当对等方 A 请求对等方 B 发送一个特定的文件时，在这个特定的通信会话中对等方 A 是客户，而对等方 B 是服务器。在不致混淆的情况下，我们有时也使用术语“应用程序的客户端和服务器端”。在本章的结尾，我们将逐步讲解网络应用程序的客户端和服务器端的简单代码。

2. 进程与计算机网络之间的接口

如上所述，多数应用程序是由通信进程对组成，每对中的两个进程互相发送报文。从一个进程向另一个进程发送的报文必须通过下面的网络。进程通过一个称为**套接字**（socket）的软件接口向网络发送报文和从网络接收报文。我们考虑一个类比来帮助我们理解进程和套接字。进程可类比于一座房子，而它的套接字可以类比于它的门。当一个进程想向位于另外一台主机上的另一个进程发送报文时，它把报文推出该门（套接字）。该发送进程假定该门到另外一侧之间有运输的基础设施，该设施将把报文传送到目的进程的门口。一旦该报文抵达目的主机，它通过接收进程的门（套接字）传递，然后接收进程对该报文进行处理。

图 2-3 显示了两个经过因特网通信的进程之间的套接字通信（图 2-3 中假定由该进程使用的下面运输层协议是因特网的 TCP 协议）。如该图所示，套接字是同一台主机内应用层与运输层之间的接口。由于该套接字是建立网络应用程序的可编程接口，因此套接字也称为应用程序和网络之间的**应用程序编程接口**（Application Programming Interface，API）。应用程序开发者可以控制套接字在应用层端的一切，但是对该套接字的运输层端几乎没有控制权。应用程序开发者对于运输层的控制仅限于：①选择运输层协议；②也许能设定几个运输层参数，如最大缓存和最大报文段长度等（将在第 3 章中涉及）。一旦应用程序开发者选择了一个运输层协议（如果可供选择的话），则应用程序就建立在由该协议提供的运输层服务之上。我们将在 2.7 节中对套接字进行更为详细的探讨。

3. 进程寻址

为了向特定目的地发送邮政邮件，目的地需要有一个地址。类似地，在一台主机上运行的进程为了向在另一台主机上运行的进程发送分组，接收进程需要有一个地址。为了标识该接收进程，需要定义两种信息：①主机的地址；②在目的主机中指定接收进程的标识符。

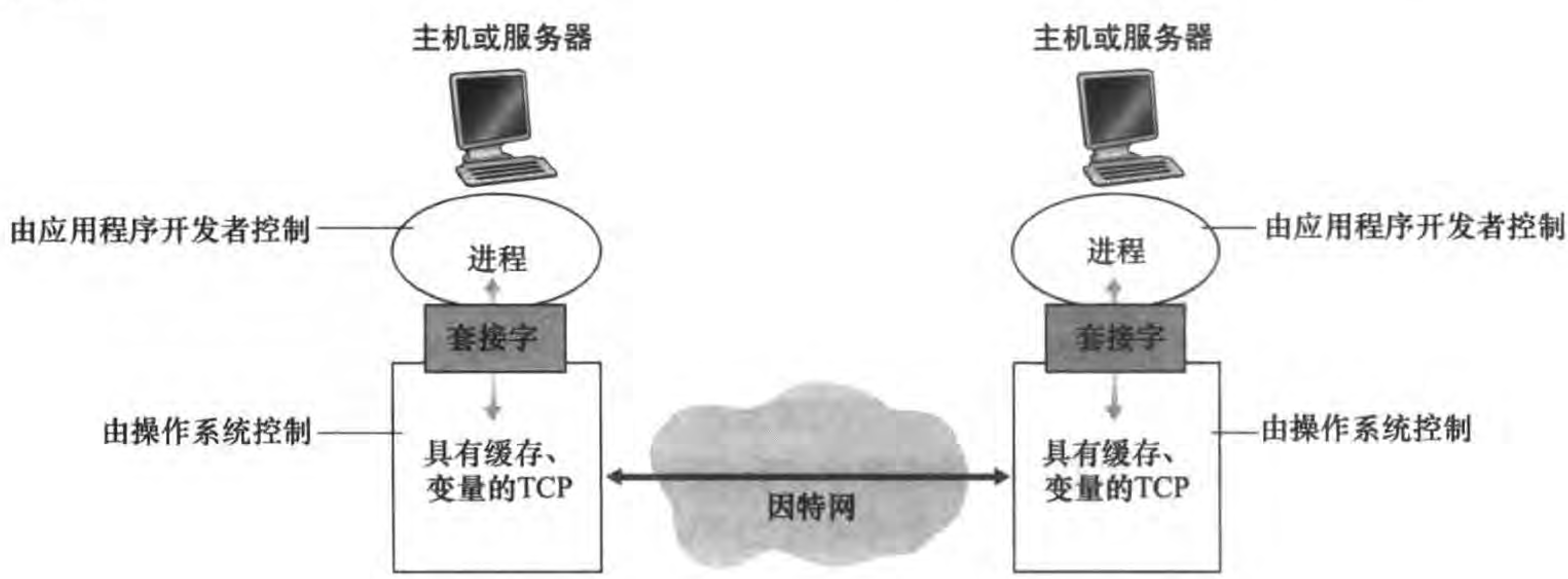


图 2-3　应用进程、套接字和下面的运输层协议

在因特网中，主机由其 **IP 地址**（IP address）标识。我们将在第 4 章中非常详细地讨论 IP 地址。此时，我们只要知道 IP 地址是一个 32 比特的量且它能够唯一地标识该主机就够了。除了知道报文发送目的地的主机地址外，发送进程还必须指定运行在接收主机上的接收进程（更具体地说，接收套接字）。因为一般而言一台主机能够运行许多网络应用，这些信息是需要的。目的地**端口号**（port number）用于这个目的。已经给流行的应用分配了特定的端口号。例如，Web 服务器用端口号 80 来标识。邮件服务器进程（使用 SMTP 协议）用端口号 25 来标识。用于所有因特网标准协议的周知端口号的列表能够在 http://www.iana.org 处找到。我们将在第 3 章中详细学习端口号。

2.1.3 可供应用程序使用的运输服务

前面讲过套接字是应用程序进程和运输层协议之间的接口。在发送端的应用程序将报文推进该套接字。在该套接字的另一侧，运输层协议负责从接收进程的套接字得到该报文。

包括因特网在内的很多网络提供了不止一种运输层协议。当开发一个应用时，必须选择一种可用的运输层协议。如何做出这种选择呢？最可能的方式是，通过研究这些可用的运输层协议所提供的服务，选择一个最能为你的应用需求提供恰当服务的协议。这种情况类似于在两个城市间旅行时选择飞机还是火车作为交通工具。每种运输模式为你提供不同的服务，你必须选择一种或另一种（例如，火车可以直到市区上客和下客，而飞机提供了更短的旅行时间）。

一个运输层协议能够为调用它的应用程序提供什么样的服务呢？我们大体能够从四个方面对应用程序服务要求进行分类：可靠数据传输、吞吐量、定时和安全性。

1. 可靠数据传输

如第 1 章讨论的那样，分组在计算机网络中可能丢失。例如，分组能够使路由器中的缓存溢出，或者当分组中的某些比特损坏后可能被丢弃。像电子邮件、文件传输、远程主机访问、Web 文档传输以及金融应用等这样的应用，数据丢失可能会造成灾难性的后果（在后一种情况下，无论对银行或对顾客都是如此！）。因此，为了支持这些应用，必须做一些工作以确保由应用程序的一端发送的数据正确、完全地交付给该应用程序的另一端。如果一个协议提供了这样的确保数据交付服务，就认为提供了**可靠数据传输**（reliable data transfer）。运输层协议能够潜在地向应用程序提供的一个重要服务是进程到进程的可靠数据传输。当一个运输协议提供这种服务时，发送进程只要将其数据传递进套接字，就可以完全相信该数据将能无差错地到达接收进程。

当一个运输层协议不提供可靠数据传输时，由发送进程发送的某些数据可能到达不了接收进程。这可能能被**容忍丢失的应用**（loss-tolerant application）所接受，最值得注意的是多媒体应用，如交谈式音频/视频，它们能够承受一定量的数据丢失。在这些多媒体应用中，丢失的数据引起播放的音频/视频出现小干扰，而不是致命的损伤。

2. 吞吐量

在第 1 章中我们引入了可用吞吐量的概念，在沿着一条网络路径上的两个进程之间的通信会话场景中，可用吞吐量就是发送进程能够向接收进程交付比特的速率。因为其他会话将共享沿着该网络路径的带宽，并且因为这些会话将会到达和离开，该可用吞吐量将随时间波动。这些观察导致另一种自然的服务，即运输层协议能够以某种特定的速率提供确

保的可用吞吐量。使用这种服务，该应用程序能够请求 r 比特/秒的确保吞吐量，并且该运输协议能够确保可用吞吐量总是为至少 r 比特/秒。这样的确保吞吐量的服务将对许多应用程序有吸引力。例如，如果因特网电话应用程序对语音以 32kbps 的速率进行编码，那么它需要以这个速率向网络发送数据，并以该速率向接收应用程序交付数据。如果运输协议不能提供这种吞吐量，该应用程序或以较低速率进行编码（并且接收足够的吞吐量以维持这种较低的编码速率），或它可能必须放弃发送，这是因为对于这种因特网电话应用而言，接收所需吞吐量的一半是几乎没有或根本没有用处的。具有吞吐量要求的应用程序被称为**带宽敏感的应用**（bandwidth-sensitive application）。许多当前的多媒体应用是带宽敏感的，尽管某些多媒体应用程序可能采用自适应编码技术对数字语音或视频以与当前可用带宽相匹配的速率进行编码。

带宽敏感的应用具有特定的吞吐量要求，而**弹性应用**（elastic application）能够根据当时可用的带宽或多或少地利用可供使用的吞吐量。电子邮件、文件传输以及 Web 传送都属于弹性应用。当然，吞吐量是越多越好。有一句格言说得好，钱越多越好，人越瘦越美，我们永远不会嫌吞吐量太多的！

3. 定时

运输层协议也能提供定时保证。如同具有吞吐量保证那样，定时保证能够以多种形式实现。一个保证的例子如：发送方注入进套接字中的每个比特到达接收方的套接字不迟于 100ms。这种服务将对交互式实时应用程序有吸引力，如因特网电话、虚拟环境、电话会议和多方游戏，所有这些服务为了有效性而要求数据交付有严格的时间限制（参见第 9 章 [Gauthier 1999；Ramjee 1994]）。例如，在因特网电话中，较长的时延会导致会话中出现不自然的停顿；在多方游戏和虚拟互动环境中，在做出动作并看到来自环境（如来自位于端到端连接中另一端点的玩家）的响应之间，较长的时延使得它失去真实感。对于非实时的应用，较低的时延总比较高的时延好，但对端到端的时延没有严格的约束。

4. 安全性

最后，运输协议能够为应用程序提供一种或多种安全性服务。例如，在发送主机中，运输协议能够加密由发送进程传输的所有数据，在接收主机中，运输层协议能够在将数据交付给接收进程之前解密这些数据。这种服务将在发送和接收进程之间提供机密性，以防该数据以某种方式在这两个进程之间被观察到。运输协议还能提供除了机密性以外的其他安全性服务，包括数据完整性和端点鉴别，我们将在第 8 章中详细讨论这些主题。

2.1.4　因特网提供的运输服务

至此，我们已经考虑了计算机网络能够提供的通用运输服务。现在我们要更为具体地考察由因特网提供的运输服务类型。因特网（更一般的是 TCP/IP 网络）为应用程序提供两个运输层协议，即 UDP 和 TCP。当你（作为一个软件开发者）为因特网创建一个新的应用时，首先要做出的决定是，选择 UDP 还是选择 TCP。每个协议为调用它们的应用程序提供了不同的服务集合。图 2-4 显示了某些所选的应用程序的服务要求。

1. TCP 服务

TCP 服务模型包括面向连接服务和可靠数据传输服务。当某个应用程序调用 TCP 作为其运输协议时，该应用程序就能获得来自 TCP 的这两种服务。

应用	数据丢失	带宽	时间敏感
文件传输	不能丢失	弹性	不
电子邮件	不能丢失	弹性	不
Web 文档	不能丢失	弹性（几 kbps）	不
因特网电话/视频会议	容忍丢失	音频（几 kbps ~ 1Mbps） 视频（10kbps ~ 5Mbps）	是，100ms
流式存储音频/视频	容忍丢失	同上	是，几秒
交互式游戏	容忍丢失	几 kbps ~ 10kbps	是，100ms
智能手机讯息	不能丢失	弹性	是和不是

图 2-4 选择的网络应用的要求

- *面向连接的服务*：在应用层数据报文开始流动之前，TCP 让客户和服务器互相交换运输层控制信息。这个所谓的握手过程提醒客户和服务器，让它们为大量分组的到来做好准备。在握手阶段后，一个 **TCP 连接**（TCP connection）就在两个进程的套接字之间建立了。这条连接是全双工的，即连接双方的进程可以在此连接上同时进行报文收发。当应用程序结束报文发送时，必须拆除该连接。在第 3 章中我们将详细讨论面向连接的服务，并分析它是如何实现的。
- *可靠的数据传送服务*：通信进程能够依靠 TCP，无差错、按适当顺序交付所有发送的数据。当应用程序的一端将字节流传进套接字时，它能够依靠 TCP 将相同的字节流交付给接收方的套接字，而没有字节的丢失和冗余。

TCP 协议还具有拥塞控制机制，这种服务不一定能为通信进程带来直接好处，但能为因特网带来整体好处。当发送方和接收方之间的网络出现拥塞时，TCP 的拥塞控制机制会抑制发送进程（客户或服务器）。如我们将在第 3 章中所见，TCP 拥塞控制也试图限制每个 TCP 连接，使它们达到公平共享网络带宽的目的。

关注安全性

TCP 安全

无论 TCP 还是 UDP 都没有提供任何加密机制，这就是说发送进程传进其套接字的数据，与经网络传送到目的进程的数据相同。因此，举例来说如果某发送进程以明文方式（即没有加密）发送了一个口令进入它的套接字，该明文口令将经过发送方与接收方之间的所有链路传送，这就可能在任何中间链路被嗅探和发现。因为隐私和其他安全问题对许多应用而言已经成为至关重要的问题，所以因特网界已经研制了 TCP 的加强版本，称为**安全套接字层**（Secure Sockets Layer，SSL）。用 SSL 加强后的 TCP 不仅能够做传统的 TCP 所能做的一切，而且提供了关键的进程到进程的安全性服务，包括加密、数据完整性和端点鉴别。我们强调 SSL 不是与 TCP 和 UDP 在相同层次上的第三种因特网运输协议，而是一种对 TCP 的加强，这种强化是在应用层上实现的。特别是，如果一个应用程序要使用 SSL 的服务，它需要在该应用程序的客户端和服务器端包括 SSL 代码（利用现有的、高度优化的库和类）。SSL 有它自己的套接字 API，这类似于传统的 TCP 套接字 API。当一个应用使用 SSL 时，发送进程向 SSL 套接字传递明文数据；在发送主

机中的 SSL 则加密该数据并将加密的数据传递给 TCP 套接字。加密的数据经因特网传送到接收进程中的 TCP 套接字。该接收套接字将加密数据传递给 SSL，由其进行解密。最后，SSL 通过它的 SSL 套接字将明文数据传递给接收进程。我们将在第 8 章中更为详细地讨论 SSL。

2. UDP 服务

UDP 是一种不提供不必要服务的轻量级运输协议，它仅提供最小服务。UDP 是无连接的，因此在两个进程通信前没有握手过程。UDP 协议提供一种不可靠数据传送服务，也就是说，当进程将一个报文发送进 UDP 套接字时，UDP 协议并不保证该报文将到达接收进程。不仅如此，到达接收进程的报文也可能是乱序到达的。

UDP 没有包括拥塞控制机制，所以 UDP 的发送端可以用它选定的任何速率向其下层（网络层）注入数据。（然而，值得注意的是实际端到端吞吐量可能小于该速率，这可能是因为中间链路的带宽受限或因为拥塞而造成的。）

3. 因特网运输协议所不提供的服务

我们已经从 4 个方面组织了运输协议服务：可靠数据传输、吞吐量、定时和安全性。TCP 和 UDP 提供了这些服务中的哪些呢？我们已经注意到 TCP 提供了可靠的端到端数据传送。并且我们也知道 TCP 在应用层可以很容易地用 SSL 来加强以提供安全服务。但在我们对 TCP 和 UDP 的简要描述中，明显地漏掉了对吞吐量或定时保证的讨论，即这些服务目前的因特网运输协议并没有提供。这是否意味着诸如因特网电话这样的时间敏感应用不能运行在今天的因特网上呢？答案显然是否定的，因为在因特网上运行时间敏感应用已经有多年了。这些应用经常工作得相当好，因为它们已经被设计成尽最大可能对付这种保证的缺乏。我们将在第 9 章中研究几种设计技巧。无论如何，在时延过大或端到端吞吐量受限时，好的设计也是有限制的。总之，今天的因特网通常能够为时间敏感应用提供满意的服务，但它不能提供任何定时或带宽保证。

图 2-5 指出了一些流行的因特网应用所使用的运输协议。可以看到，电子邮件、远程终端访问、Web、文件传输都使用了 TCP。这些应用选择 TCP 的最主要原因是 TCP 提供了可靠数据传输服务，确保所有数据最终到达目的地。因为因特网电话应用（如 Skype）通常能够容忍某些丢失但要求达到一定的最小速率才能有效工作，所以因特网电话应用的开发者通常愿意将该应用运行在 UDP 上，从而设法避开 TCP 的拥塞控制机制和分组开销。但因为许多防火墙被配置成阻挡（大多数类型的）UDP 流量，所以因特网电话应用通常设计成如果 UDP 通信失败就使用 TCP 作为备份。

应用	应用层协议	支撑的运输协议
电子邮件	SMTP [RFC 5321]	TCP
远程终端访问	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
文件传输	FTP [RFC 959]	TCP
流式多媒体	HTTP（如 YouTube）	TCP
因特网电话	SIP [RFC 3261]、RTP [RFC 3550] 或专用的（如 Skype）	UDP 或 TCP

图 2-5　流行的因特网应用及其应用层协议和支撑的运输协议

2.1.5 应用层协议

我们刚刚学习了通过把报文发送进套接字实现网络进程间的相互通信。但是如何构造这些报文？在这些报文中的各个字段的含义是什么？进程何时发送这些报文？这些问题将我们带进应用层协议的范围。**应用层协议**（application-layer protocol）定义了运行在不同端系统上的应用程序进程如何相互传递报文。特别是应用层协议定义了：

- 交换的报文类型，例如请求报文和响应报文。
- 各种报文类型的语法，如报文中的各个字段及这些字段是如何描述的。
- 字段的语义，即这些字段中的信息的含义。
- 确定一个进程何时以及如何发送报文，对报文进行响应的规则。

有些应用层协议是由 RFC 文档定义的，因此它们位于公共域中。例如，Web 的应用层协议 HTTP（超文本传输协议 [RFC 2616]）就作为一个 RFC 可供使用。如果浏览器开发者遵从 HTTP RFC 规则，所开发出的浏览器就能访问任何遵从该文档标准的 Web 服务器并获取相应 Web 页面。还有很多别的应用层协议是专用的，有意不为公共域使用。例如，Skype 使用了专用的应用层协议。

区分网络应用和应用层协议是很重要的。应用层协议只是网络应用的一部分（尽管从我们的角度看，它是应用非常重要的一部分）。我们来看一些例子。Web 是一种客户 - 服务器应用，它允许客户按照需求从 Web 服务器获得文档。该 Web 应用有很多组成部分，包括文档格式的标准（即 HTML）、Web 浏览器（如 Firefox 和 Microsoft Internet Explorer）、Web 服务器（如 Apache、Microsoft 服务器程序），以及一个应用层协议。Web 的应用层协议是 HTTP，它定义了在浏览器和 Web 服务器之间传输的报文格式和序列。因此，HTTP 只是 Web 应用的一个部分（尽管是重要部分）。举另外一个例子，因特网电子邮件应用也有很多组成部分，包括能容纳用户邮箱的邮件服务器、允许用户读取和生成邮件的邮件客户程序（如 Microsoft Outlook）、定义电子邮件报文结构的标准、定义报文如何在服务器之间以及如何在服务器与邮件客户程序之间传递的应用层协议、定义如何对报文首部的内容进行解释的应用层协议。用于电子邮件的主要应用层协议就是 SMTP（简单邮件传输协议 [RFC5321]）。因此，电子邮件的首要应用层协议 SMTP 也只是电子邮件应用的一个部分（尽管是重要部分）。

2.1.6 本书涉及的网络应用

每天都有新的公共域或者专用域因特网应用被开发出来。我们不愿像百科全书一样涉及大量的因特网应用，而是选择其中几种重要而流行的应用加以关注。在本章中我们详细讨论 5 种重要的应用：Web、文件传输、电子邮件、目录服务、流式视频和 P2P。我们首先讨论 Web 应用，不仅因为它是极为流行的应用，而且因为它的应用层协议 HTTP比较简单并且易于理解。我们接下来讨论电子邮件，这是因特网上第一个招人喜爱的应用程序。说电子邮件比 Web 更复杂，是因为它使用了多个而不是一个应用层协议。在电子邮件之后，我们学习 DNS，它为因特网提供目录服务。大多数用户不直接与 DNS 打交道，而是通过其他的应用（包括 Web、文件传输和电子邮件）间接使用它。DNS 很好地说明了一种核心的网络功能（网络名字到网络地址的转换）是怎样在因特网的应用层实现的。然后我们讨论 P2P 文件共享应用，通过讨论包括经内容分发网分发存储的视频在内的按需流式视频，结束应用层的学习。在第 9 章中，我们将涉及多媒体

应用，包括 VoIP 视频会议。

2.2　Web 和 HTTP

20 世纪 90 年代以前，因特网的主要使用者还是研究人员、学者和大学生，他们登录远程主机，在本地主机和远程主机之间传输文件，收发新闻，收发电子邮件。尽管这些应用非常有用（并且继续如此），但是因特网基本上不为学术界和研究界之外所知。到了 20 世纪 90 年代初期，一个主要的新型应用即万维网（World Wide Web）登上了舞台［Berners-Lee 1994］。Web 是一个引起公众注意的因特网应用，它极大地改变了人们与工作环境内外交流的方式。它将因特网从只是很多数据网之一的地位提升为仅有的一个数据网。

也许对大多数用户来说，最具有吸引力的就是 Web 的按需操作。当用户需要时，就能得到所想要的内容。这不同于无线电广播和电视，它们迫使用户只能收听、收看内容提供者提供的节目。除了可以按需操作以外，Web 还有很多让人们喜欢和珍爱的特性。任何人使信息在 Web 上可用都非常简单，即只需要极低的费用就能成为出版人。超链接和搜索引擎帮助我们在 Web 站点的海洋里导航。图片和视频刺激着我们的感官。表单、JavaScript、Java 小程序和很多其他的装置，使我们可以与 Web 页面和站点进行交互。并且，Web 及其协议作为平台，为 YouTube、基于 Web 的电子邮件（如 Gmail）和大多数移动因特网应用（包括 Instagram 和谷歌地图）服务。

2.2.1　HTTP 概况

Web 的应用层协议是**超文本传输协议**（HyperText Transfer Protocol，HTTP），它是 Web 的核心，在［RFC 1945］和［RFC 2616］中进行了定义。HTTP 由两个程序实现：一个客户程序和一个服务器程序。客户程序和服务器程序运行在不同的端系统中，通过交换 HTTP 报文进行会话。HTTP 定义了这些报文的结构以及客户和服务器进行报文交换的方式。在详细解释 HTTP 之前，应当回顾某些 Web 术语。

Web 页面（Web page）（也叫文档）是由对象组成的。一个**对象**（object）只是一个文件，诸如一个 HTML 文件、一个 JPEG 图形、一个 Java 小程序或一个视频片段这样的文件，且它们可通过一个 URL 地址寻址。多数 Web 页面含有一个 **HTML 基本文件**（base HTML file）以及几个引用对象。例如，如果一个 Web 页面包含 HTML 文本和 5 个 JPEG 图形，那么这个 Web 页面有 6 个对象：一个 HTML 基本文件加 5 个图形。HTML 基本文件通过对象的 URL 地址引用页面中的其他对象。每个 URL 地址由两部分组成：存放对象的服务器主机名和对象的路径名。例如，URL 地址 http://www. someSchool. edu/someDepartment/picture. gif，其中的 www. someSchool. edu 就是主机名，/someDepartment/picture. gif 就是路径名。因为 **Web 浏览器**（Web browser）（例如 Internet Explorer 和 Firefox）实现了 HTTP 的客户端，所以在 Web 环境中我们经常交替使用“浏览器”和“客户”这两个术语。**Web 服务器**（Web server）实现了 HTTP 的服务器端，它用于存储 Web 对象，每个对象由 URL 寻址。流行的 Web 服务器有 Apache 和 Microsoft Internet Information Server（微软互联网信息服务器）。

HTTP 定义了 Web 客户向 Web 服务器请求 Web 页面的方式，以及服务器向客户传送 Web 页面的方式。我们稍后详细讨论客户和服务器的交互过程，而其基本思想在

图 2-6 中进行了图示。当用户请求一个 Web 页面（如点击一个超链接）时，浏览器向服务器发出对该页面中所包含对象的 HTTP请求报文，服务器接收到请求并用包含这些对象的 HTTP 响应报文进行响应。

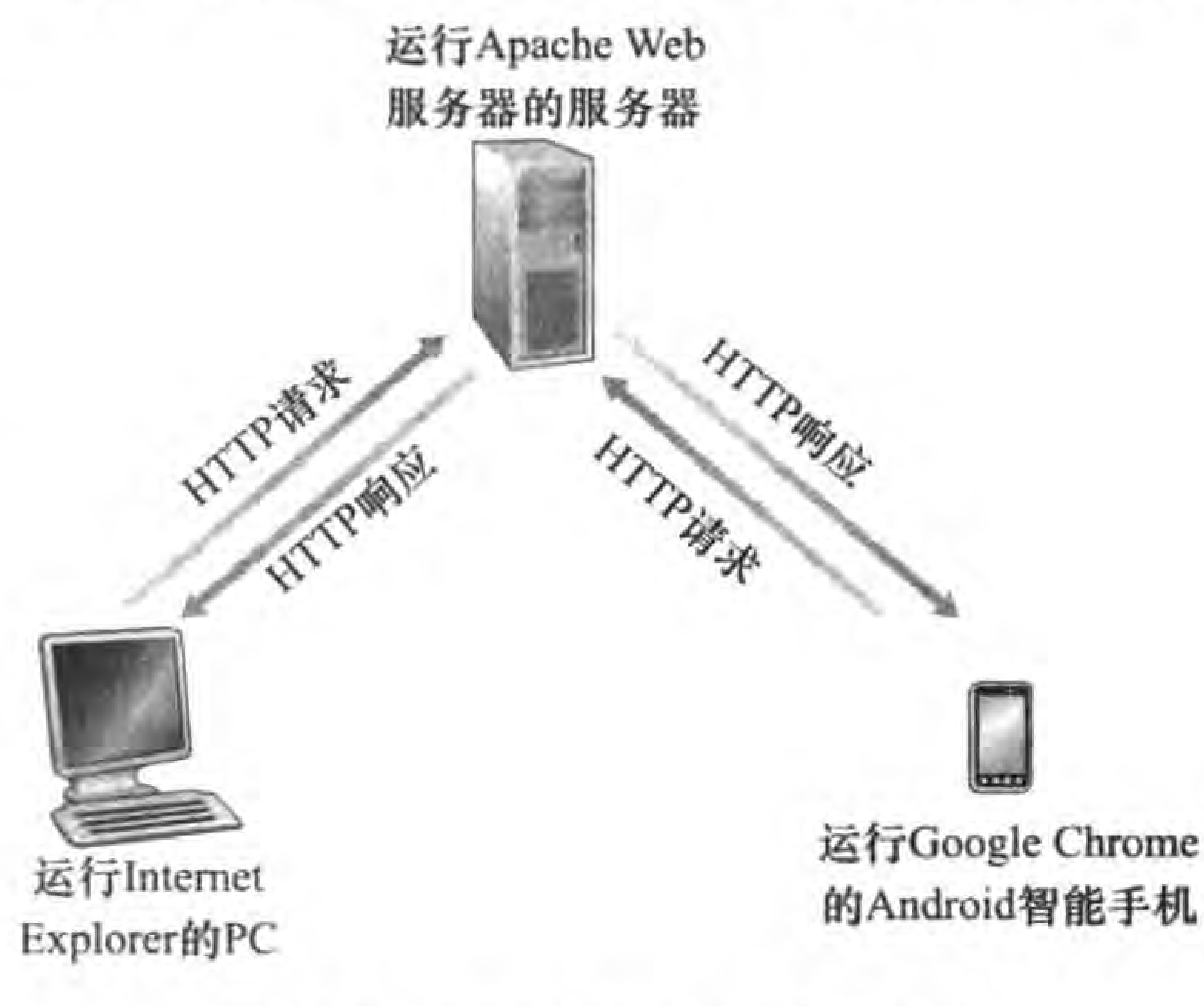


图 2-6 HTTP 的请求 - 响应行为

HTTP 使用 TCP 作为它的支撑运输协议（而不是在 UDP 上运行）。HTTP 客户首先发起一个与服务器的 TCP 连接。一旦连接建立，该浏览器和服务器进程就可以通过套接字接口访问 TCP。如同在 2.1 节中描述的那样，客户端的套接字接口是客户进程与 TCP 连接之间的门，在服务器端的套接字接口则是服务器进程与 TCP 连接之间的门。客户向它的套接字接口发送 HTTP 请求报文并从它的套接字接口接收 HTTP 响应报文。类似地，服务器从它的套接字接口接收 HTTP 请求报文和向它的套接字接口发送 HTTP 响应报文。一旦客户向它的套接字接口发送了一个请求报文，该报文就脱离了客户控制并进入 TCP 的控制。2.1 节讲过，TCP 为 HTTP 提供可靠数据传输服务。这意味着，一个客户进程发出的每个 HTTP 请求报文最终能完整地到达服务器；类似地，服务器进程发出的每个 HTTP 响应报文最终能完整地到达客户。这里我们看到了分层体系结构最大的优点，即 HTTP 协议不用担心数据丢失，也不关注 TCP 从网络的数据丢失和乱序故障中恢复的细节。那是 TCP 以及协议栈较低层协议的工作。

注意到下列现象很重要：服务器向客户发送被请求的文件，而不存储任何关于该客户的状态信息。假如某个特定的客户在短短的几秒内两次请求同一个对象，服务器并不会因为刚刚为该客户提供了该对象就不再做出反应，而是重新发送该对象，就像服务器已经完全忘记不久之前所做过的事一样。因为 HTTP 服务器并不保存关于客户的任何信息，所以我们说 HTTP 是一个**无状态协议**（stateless protocol）。我们同时也注意到 Web 使用了客户 - 服务器应用程序体系结构（如 2.1 节所述）。Web 服务器总是打开的，具有一个固定的 IP 地址，且它服务于可能来自数以百万计的不同浏览器的请求。

2.2.2 非持续连接和持续连接

在许多因特网应用程序中，客户和服务器在一个相当长的时间范围内通信，其中客户发出一系列请求并且服务器对每个请求进行响应。依据应用程序以及该应用程序的使用方式，这一系列请求可以以规则的间隔周期性地或者间断性地一个接一个发出。当这种客户 - 服务器的交互是经 TCP 进行的，应用程序的研制者就需要做一个重要决定，即每个请求/响应对是经一个单独的 TCP 连接发送，还是所有的请求及其响应经相同的 TCP 连接发送呢？采用前一种方法，该应用程序被称为使用**非持续连接**（non-persistent connection）；采用后一种方法，该应用程序被称为使用**持续连接**（persistent connection）。为了深入地理解该设计问题，我们研究在特定的应用程序即 HTTP 的情况下持续连接的优点和缺点，HTTP既能够使用非持续连接，也能够使用持续连接。尽管 HTTP 在其默认方式下使用持续连接，HTTP 客户和服务器也能配置成使用非持续连接。

1. 采用非持续连接的HTTP

我们看看在非持续连接情况下，从服务器向客户传送一个Web页面的步骤。假设该页面含有一个HTML基本文件和10个JPEG图形，并且这11个对象位于同一台服务器上。进一步假设该HTML文件的URL为：http://www.someSchool.edu/someDepartment/home.index。

我们看看发生了什么情况：

1）HTTP客户进程在端口号80发起一个到服务器www.someSchool.edu的TCP连接，该端口号是HTTP的默认端口。在客户和服务器上分别有一个套接字与该连接相关联。

2）HTTP客户经它的套接字向该服务器发送一个HTTP请求报文。请求报文中包含了路径名/someDepartment/home.index（后面我们会详细讨论HTTP报文）。

3）HTTP服务器进程经它的套接字接收该请求报文，从其存储器（RAM或磁盘）中检索出对象www.someSchool.edu/someDepartment/home.index，在一个HTTP响应报文中封装对象，并通过其套接字向客户发送响应报文。

4）HTTP服务器进程通知TCP断开该TCP连接。（但是直到TCP确认客户已经完整地收到响应报文为止，它才会实际中断连接。）

5）HTTP客户接收响应报文，TCP连接关闭。该报文指出封装的对象是一个HTML文件，客户从响应报文中提取出该文件，检查该HTML文件，得到对10个JPEG图形的引用。

6）对每个引用的JPEG图形对象重复前4个步骤。

当浏览器收到Web页面后，向用户显示该页面。两个不同的浏览器也许会以不同的方式解释（即向用户显示）该页面。HTTP与客户如何解释一个Web页面毫无关系。HTTP规范（[RFC 1945]和[RFC 2616]）仅定义了在HTTP客户程序与HTTP服务器程序之间的通信协议。

上面的步骤举例说明了非持续连接的使用，其中每个TCP连接在服务器发送一个对象后关闭，即该连接并不为其他的对象而持续下来。值得注意的是每个TCP连接只传输一个请求报文和一个响应报文。因此在本例中，当用户请求该Web页面时，要产生11个TCP连接。

在上面描述的步骤中，我们有意没有明确客户获得这10个JPEG图形对象是使用10个串行的TCP连接，还是某些JPEG对象使用了一些并行的TCP连接。事实上，用户能够配置现代浏览器来控制连接的并行度。在默认方式下，大部分浏览器打开5~10个并行的TCP连接，而每条连接处理一个请求响应事务。如果用户愿意，最大并行连接数可以设置为1，这样10条连接就会串行建立。我们在下一章会看到，使用并行连接可以缩短响应时间。

在继续讨论之前，我们来简单估算一下从客户请求HTML基本文件起到该客户收到整个文件止所花费的时间。为此，我们给出**往返时间**（Round-Trip Time，RTT）的定义，该时间是指一个短分组从客户到服务器然后再返回客户所花费的时间。RTT包括分组传播时延、分组在中间路由器和交换机上的排队时延以及分组处理时延（这些时延在1.4节已经讨论过）。现在考虑当用户点击超链接时会发生什么现象。如图2-7所示，

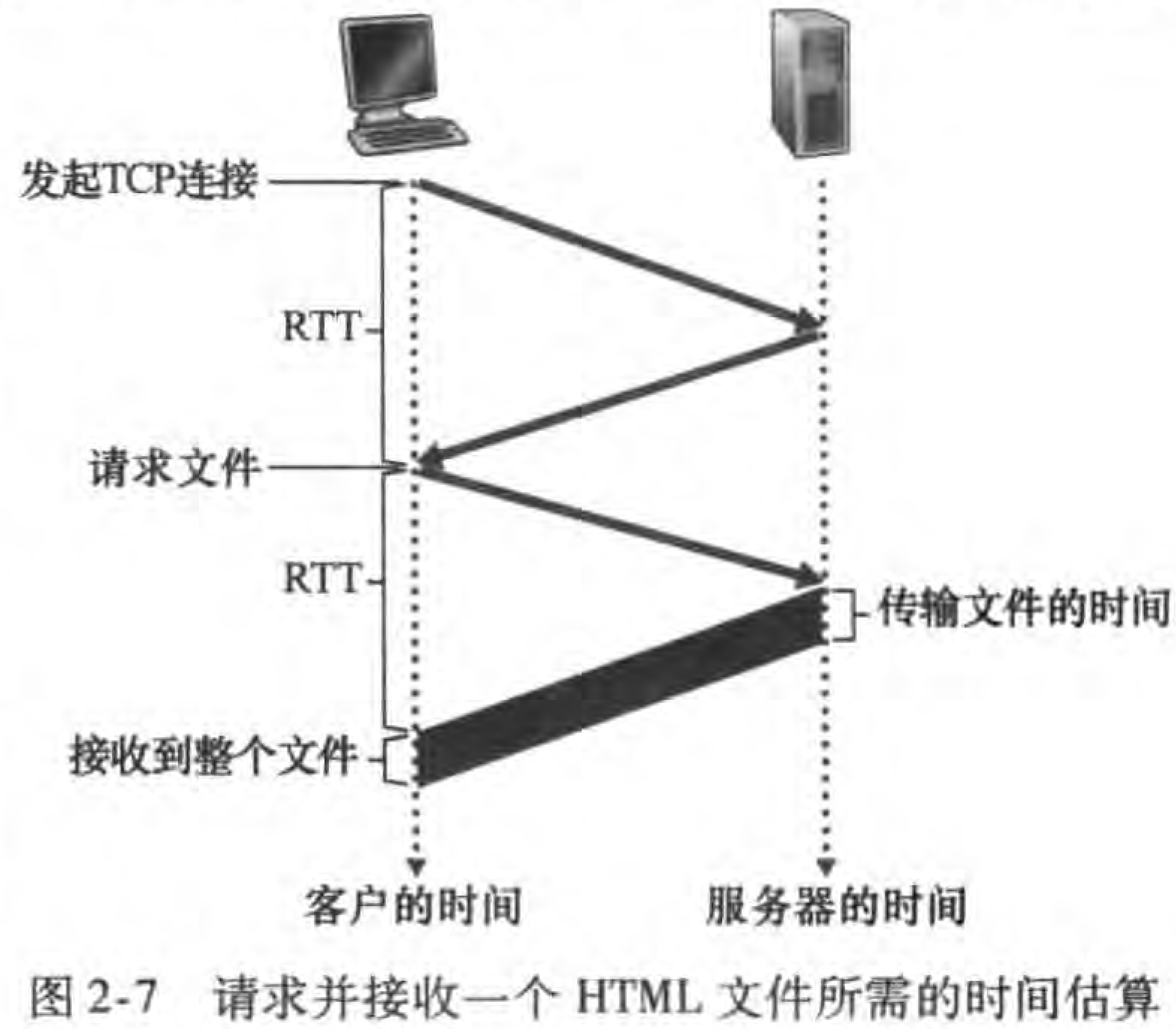


图2-7 请求并接收一个HTML文件所需的时间估算

这引起浏览器在它和 Web 服务器之间发起一个 TCP 连接；这涉及一次“三次握手”过程，即客户向服务器发送一个小 TCP 报文段，服务器用一个小 TCP 报文段做出确认和响应，最后，客户向服务器返回确认。三次握手中前两个部分所耗费的时间占用了一个 RTT。完成了三次握手的前两个部分后，客户结合三次握手的第三部分（确认）向该 TCP 连接发送一个 HTTP 请求报文。一旦该请求报文到达服务器，服务器就在该 TCP 连接上发送 HTML文件。该 HTTP 请求/响应用去了另一个 RTT。因此，粗略地讲，总的响应时间就是两个 RTT 加上服务器传输 HTML 文件的时间。

2. 采用持续连接的 HTTP

非持续连接有一些缺点。第一，必须为每一个请求的对象建立和维护一个全新的连接。对于每个这样的连接，在客户和服务器中都要分配 TCP 的缓冲区和保持 TCP 变量，这给 Web 服务器带来了严重的负担，因为一台 Web 服务器可能同时服务于数以百计不同的客户的请求。第二，就像我们刚描述的那样，每一个对象经受两倍 RTT 的交付时延，即一个 RTT 用于创建 TCP，另一个 RTT 用于请求和接收一个对象。

在采用 HTTP 1.1 持续连接的情况下，服务器在发送响应后保持该 TCP 连接打开。在相同的客户与服务器之间，后续的请求和响应报文能够通过相同的连接进行传送。特别是，一个完整的 Web 页面（上例中的 HTML 基本文件加上 10 个图形）可以用单个持续 TCP 连接进行传送。更有甚者，位于同一台服务器的多个 Web 页面在从该服务器发送给同一个客户时，可以在单个持续 TCP 连接上进行。对对象的这些请求可以一个接一个地发出，而不必等待对未决请求（流水线）的回答。一般来说，如果一条连接经过一定时间间隔（一个可配置的超时间隔）仍未被使用，HTTP 服务器就关闭该连接。HTTP 的默认模式是使用带流水线的持续连接。最近，HTTP/2 [RFC 7540] 是在 HTTP 1.1 基础上构建的，它允许在相同连接中多个请求和回答交错，并增加了在该连接中优化 HTTP 报文请求和回答的机制。我们把量化比较持续连接和非持续连接性能的任务留作第 2、3 章的课后习题。鼓励读者阅读文献 [Heidemann 1997; Nielsen 1997; RFC 7540]。

2.2.3 HTTP 报文格式

HTTP 规范 [RFC 1945; RFC 2616; RFC 7540] 包含了对 HTTP 报文格式的定义。HTTP 报文有两种：请求报文和响应报文。下面讨论这两种报文。

1. HTTP 请求报文

下面提供了一个典型的 HTTP 请求报文：

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

通过仔细观察这个简单的请求报文，我们就能学到很多东西。首先，我们看到该报文是用普通的 ASCII 文本书写的，这样有一定计算机知识的人都能够阅读它。其次，我们看到该报文由 5 行组成，每行由一个回车和换行符结束。最后一行后再附加一个回车换行符。虽然这个特定的报文仅有 5 行，但一个请求报文能够具有更多的行或者至少为一行。HTTP 请求报文的第一行叫作**请求行**（request line），其后继的行叫作**首部行**（header line）。请求行有 3 个字段：方法字段、URL 字段和 HTTP 版本字段。方法字段可以取几种

不同的值，包括GET、POST、HEAD、PUT和DELETE。绝大部分的HTTP请求报文使用GET方法。当浏览器请求一个对象时，使用GET方法，在URL字段带有请求对象的标识。在本例中，该浏览器正在请求对象/somedir/page.html。其版本字段是自解释的；在本例中，浏览器实现的是HTTP/1.1版本。

现在我们看看本例的首部行。首部行Host：www.someschool.edu指明了对象所在的主机。你也许认为该首部行是不必要的，因为在该主机中已经有一条TCP连接存在了。但是，如我们将在2.2.5节中所见，该首部行提供的信息是Web代理高速缓存所要求的。通过包含Connection：close首部行，该浏览器告诉服务器不要麻烦地使用持续连接，它要求服务器在发送完被请求的对象后就关闭这条连接。User-agent：首部行用来指明用户代理，即向服务器发送请求的浏览器的类型。这里浏览器类型是Mozilla/5.0，即Firefox浏览器。这个首部行是有用的，因为服务器可以有效地为不同类型的用户代理实际发送相同对象的不同版本。(每个版本都由相同的URL寻址。) 最后，Accept-language：首部行表示用户想得到该对象的法语版本（如果服务器中有这样的对象的话）；否则，服务器应当发送它的默认版本。Accept-language：首部行仅是HTTP中可用的众多内容协商首部之一。

看过一个例子之后，我们再来看看如图2-8所示的一个请求报文的通用格式。我们看到该通用格式与我们前面的例子密切对应。然而，你可能已经注意到了在首部行（和附加的回车和换行）后有一个"实体体"（entity body）。使用GET方法时实体体为空，而使用POST方法时才使用该实体体。当用户提交表单时，HTTP客户常常使用POST方法，例如当用户向搜索引擎提供搜索关键词时。使用POST报文时，用户仍可以向服务器请求一个Web页面，但Web页面的特定内容依赖于用户在表单字段中输入的内容。如果方法字段的值为POST时，则实体体中包含的就是用户在表单字段中的输入值。

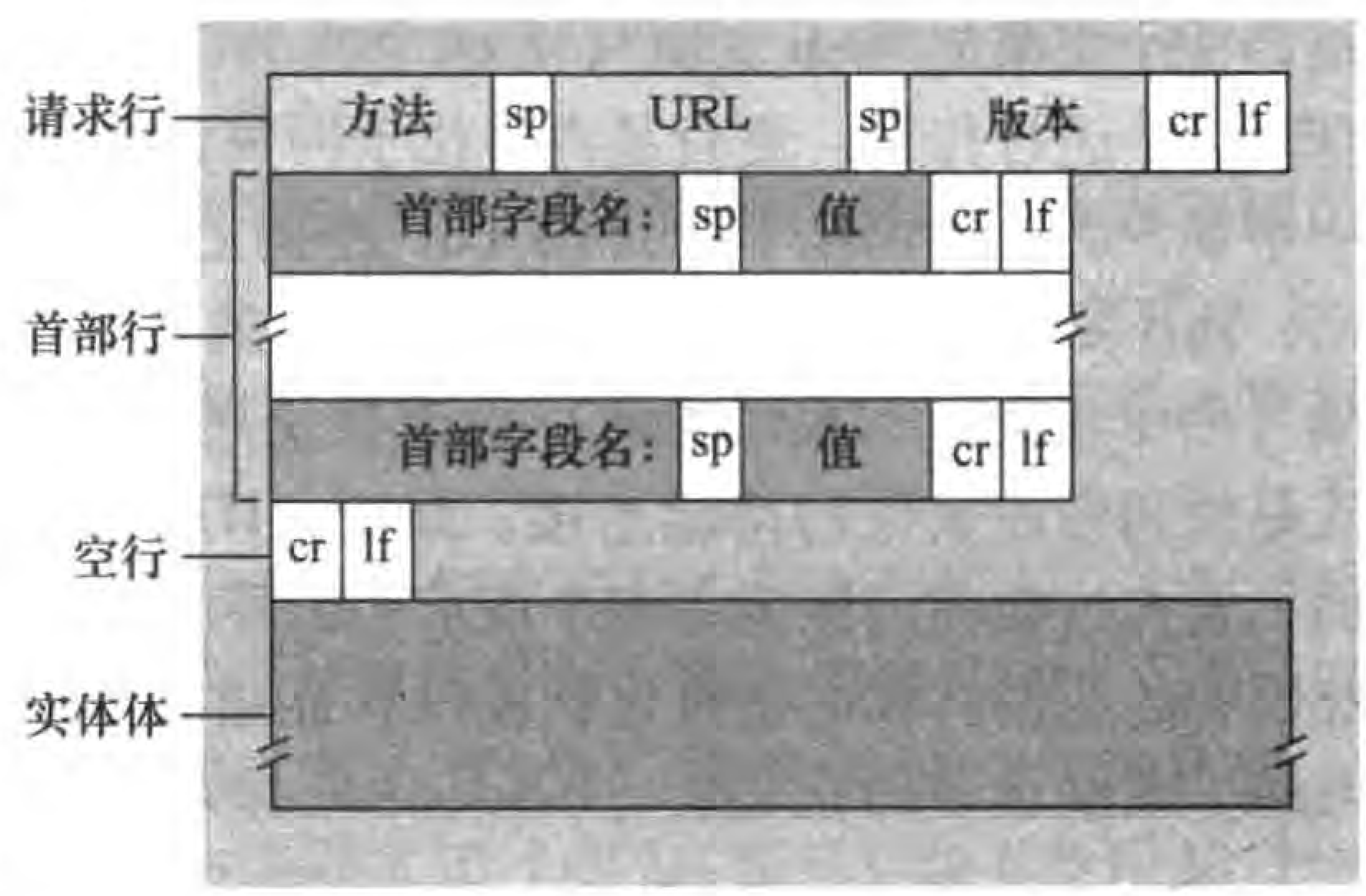


图2-8 一个HTTP请求报文的通用格式

当然，如果不提"用表单生成的请求报文不是必须使用POST方法"这一点，那将是失职。相反，HTML表单经常使用GET方法，并在（表单字段中）所请求的URL中包括输入的数据。例如，一个表单使用GET方法，它有两个字段，分别填写的是"monkeys"和"bananas"，这样，该URL结构为www.somesite.com/animalsearch?monkeys&bananas。在日复一日的网上冲浪中，你也许已经留意到了这种扩展的URL。

HEAD方法类似于GET方法。当服务器收到一个使用HEAD方法的请求时，将会用一个HTTP报文进行响应，但是并不返回请求对象。应用程序开发者常用HEAD方法进行调试跟踪。PUT方法常与Web发行工具联合使用，它允许用户上传对象到指定的Web服务器上指定的路径（目录）。PUT方法也被那些需要向Web服务器上传对象的应用程序使用。DELETE方法允许用户或者应用程序删除Web服务器上的对象。

2. HTTP响应报文

下面我们提供了一条典型的HTTP响应报文。该响应报文可以是对刚刚讨论的例子中

请求报文的响应。

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

我们仔细看一下这个响应报文。它有三个部分：一个初始**状态行**（status line），6 个**首部行**（header line），然后是**实体体**（entity body）。实体体部分是报文的主要部分，即它包含了所请求的对象本身（表示为 data data data data data …）。状态行有 3 个字段：协议版本字段、状态码和相应状态信息。在这个例子中，状态行指示服务器正在使用 HTTP/1.1，并且一切正常（即服务器已经找到并正在发送所请求的对象）。

我们现在来看看首部行。服务器用 Connection：close 首部行告诉客户，发送完报文后将关闭该 TCP 连接。Date：首部行指示服务器产生并发送该响应报文的日期和时间。值得一提的是，这个时间不是指对象创建或者最后修改的时间，而是服务器从它的文件系统中检索到该对象，将该对象插入响应报文，并发送该响应报文的时间。Server：首部行指示该报文是由一台 Apache Web 服务器产生的，它类似于 HTTP 请求报文中的 User-agent：首部行。Last-Modified：首部行指示了对象创建或者最后修改的日期和时间。Last-Modified：首部行对既可能在本地客户也可能在网络缓存服务器上的对象缓存来说非常重要。我们将很快详细地讨论缓存服务器（也叫代理服务器）。Content-Length：首部行指示了被发送对象中的字节数。Content-Type：首部行指示了实体体中的对象是 HTML 文本。（该对象类型应该正式地由 Content-Type：首部行而不是用文件扩展名来指示。）

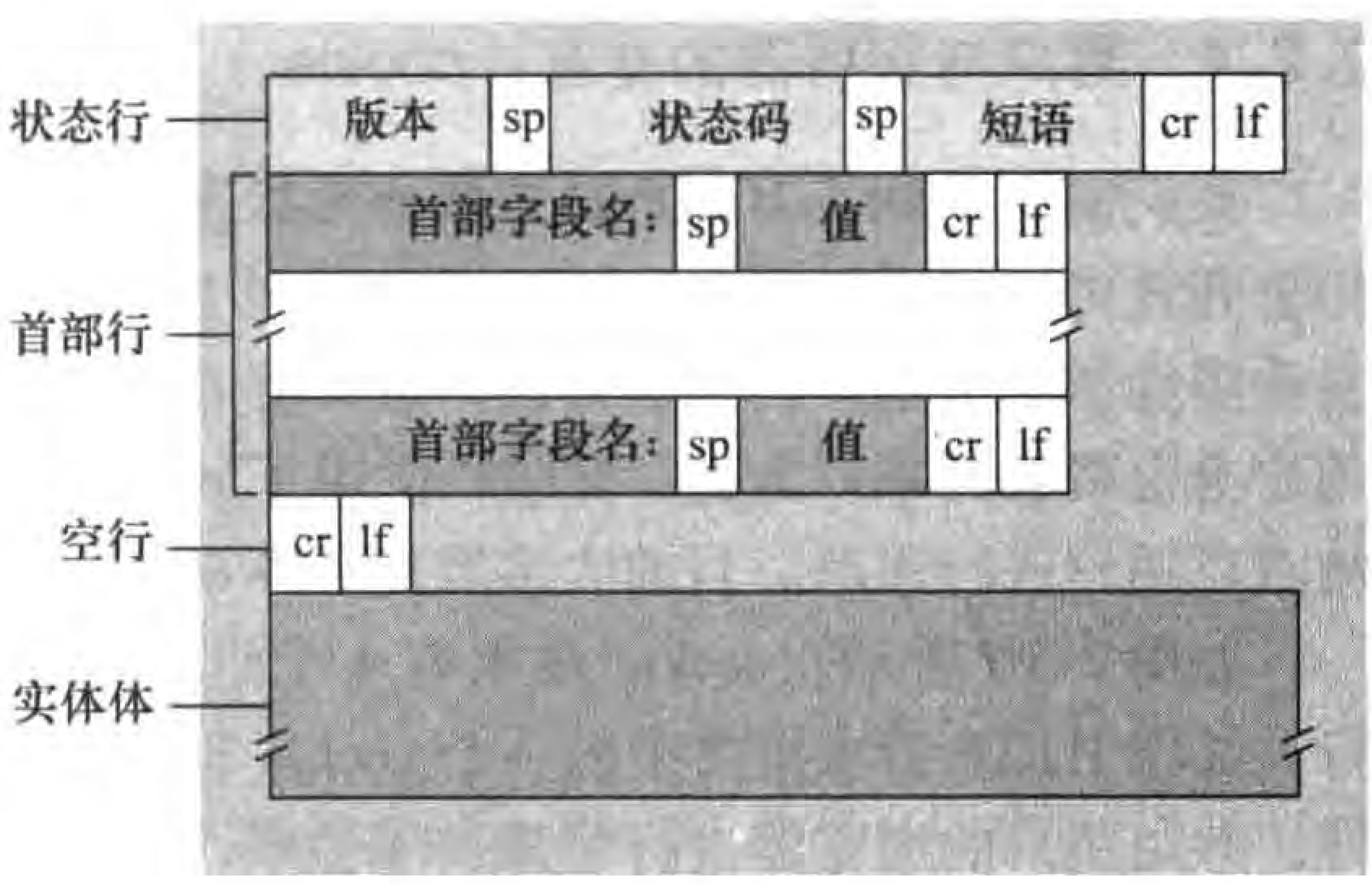


图 2-9 一个 HTTP 响应报文的通用格式

看过一个例子后，我们再来查看响应报文的通用格式，如图 2-9 所示。该通用格式与前面例子中的响应报文相匹配。我们补充说明一下状态码和它们对应的短语。状态码及其相应的短语指示了请求的结果。一些常见的状态码和相关的短语包括：

- 200 OK：请求成功，信息在返回的响应报文中。
- 301 Moved Permanently：请求的对象已经被永久转移了，新的 URL 定义在响应报文的 Location：首部行中。客户软件将自动获取新的 URL。
- 400 Bad Request：一个通用差错代码，指示该请求不能被服务器理解。
- 404 Not Found：被请求的文档不在服务器上。
- 505 HTTP Version Not Supported：服务器不支持请求报文使用的 HTTP 协议版本。

你想看一下真实的 HTTP 响应报文吗？这正是我们高度推荐而且也很容易做到的事。首先用 Telnet 登录到你喜欢的 Web 服务器上，接下来输入一个只有一行的请求报文去请求放在该服务器上的某些对象。例如，假设你看到命令提示，键入：

```
telnet gaia.cs.umass.edu 80
GET /kurose_ross/interactive/index.php HTTP/1.1
Host: gaia.cs.umass.edu
```

（在输入最后一行后连续按两次回车。）这就打开一个到主机 gaia. cs. umass. edu 的 80 端口的 TCP 连接，并发送一个 HTTP 请求报文。你将会看到一个携带包括本书交互式课后作业的基本 HTML 文件的响应报文。如果你只是想看一下 HTTP 协议的报文行，而不是获取对象本身的话，那么可以用 HEAD 代替 GET。

在本节中，我们讨论了 HTTP 请求报文和响应报文中的一些首部行。HTTP 规范中定义了许许多多的首部行，这些首部行可以被浏览器、Web 服务器和 Web 缓存服务器插入。我们只提到了全部首部行中的少数几个，在 2.2.5 节中我们讨论网络 Web 缓存时还会涉及其他几个。一本可读性很强的文献是［Krishnamurty 2001］，它对 HTTP 协议（包括它的首部行和状态码）进行了广泛讨论。

浏览器是如何决定在一个请求报文中包含哪些首部行的呢？Web 服务器又是如何决定在一个响应报文中包含哪些首部行呢？浏览器产生的首部行与很多因素有关，包括浏览器的类型和协议版本（例如，HTTP/1.0 浏览器将不会产生任何 1.1 版本的首部行）、浏览器的用户配置（如喜好的语言）、浏览器当前是否有一个缓存的但是可能超期的对象版本。Web 服务器的表现也类似：在产品、版本和配置上都有差异，所有这些都会影响响应报文中包含的首部行。

2.2.4 用户与服务器的交互：cookie

我们前面提到了 HTTP 服务器是无状态的。这简化了服务器的设计，并且允许工程师们去开发可以同时处理数以千计的 TCP 连接的高性能 Web 服务器。然而一个 Web 站点通常希望能够识别用户，可能是因为服务器希望限制用户的访问，或者因为它希望把内容与用户身份联系起来。为此，HTTP 使用了 cookie。cookie 在［RFC 6265］中定义，它允许站点对用户进行跟踪。目前大多数商务 Web 站点都使用了 cookie。

如图 2-10 所示，cookie 技术有 4 个组件：①在 HTTP 响应报文中的一个 cookie 首部行；②在 HTTP 请求报文中的一个 cookie 首部行；③在用户端系统中保留有一个 cookie 文件，并由用户的浏览器进行管理；④位于 Web 站点的一个后端数据库。使用图 2-10，我们通过一个典型的例子看看 cookie 的工作过程。假设 Susan 总是从家中 PC 使用 Internet Explorer 上网，她首次与 Amazon. com 联系。我们假定过去她已经访问过 eBay 站点。当请求报文到达该 Amazon Web 服务器时，该 Web 站点将产生一个唯一识别码，并以此作为索引在它的后端数据库中产生一个表项。接下来 Amazon Web 服务器用一个包含 Set-cookie：首部的 HTTP 响应报文对 Susan 的浏览器进行响应，其中 Set-cookie：首部含有该识别码。例如，该首部行可能是

```
Set-cookie: 1678
```

当 Susan 的浏览器收到了该 HTTP 响应报文时，它会看到该 Set-cookie：首部。该浏览器在它管理的特定 cookie 文件中添加一行，该行包含服务器的主机名和在 Set-cookie：首部中的识别码。值得注意的是该 cookie 文件已经有了用于 eBay 的表项，因为 Susan 过去访问过该站点。当 Susan 继续浏览 Amazon 网站时，每请求一个 Web 页面，其浏览器就会查询该 cookie 文件并抽取她对这个网站的识别码，并放到 HTTP 请求报文中包括识别码的 cookie 首

部行中。特别是，发往该 Amazon 服务器的每个 HTTP 请求报文都包括以下首部行：

```
Cookie: 1678
```

在这种方式下，Amazon 服务器可以跟踪 Susan 在 Amazon 站点的活动。尽管 Amazon Web 站点不必知道 Susan 的名字，但它确切地知道用户 1678 按照什么顺序、在什么时间、访问了哪些页面！Amazon 使用 cookie 来提供它的购物车服务，即 Amazon 能够维护 Susan 希望购买的物品列表，这样在 Susan 结束会话时可以一起为它们付费。

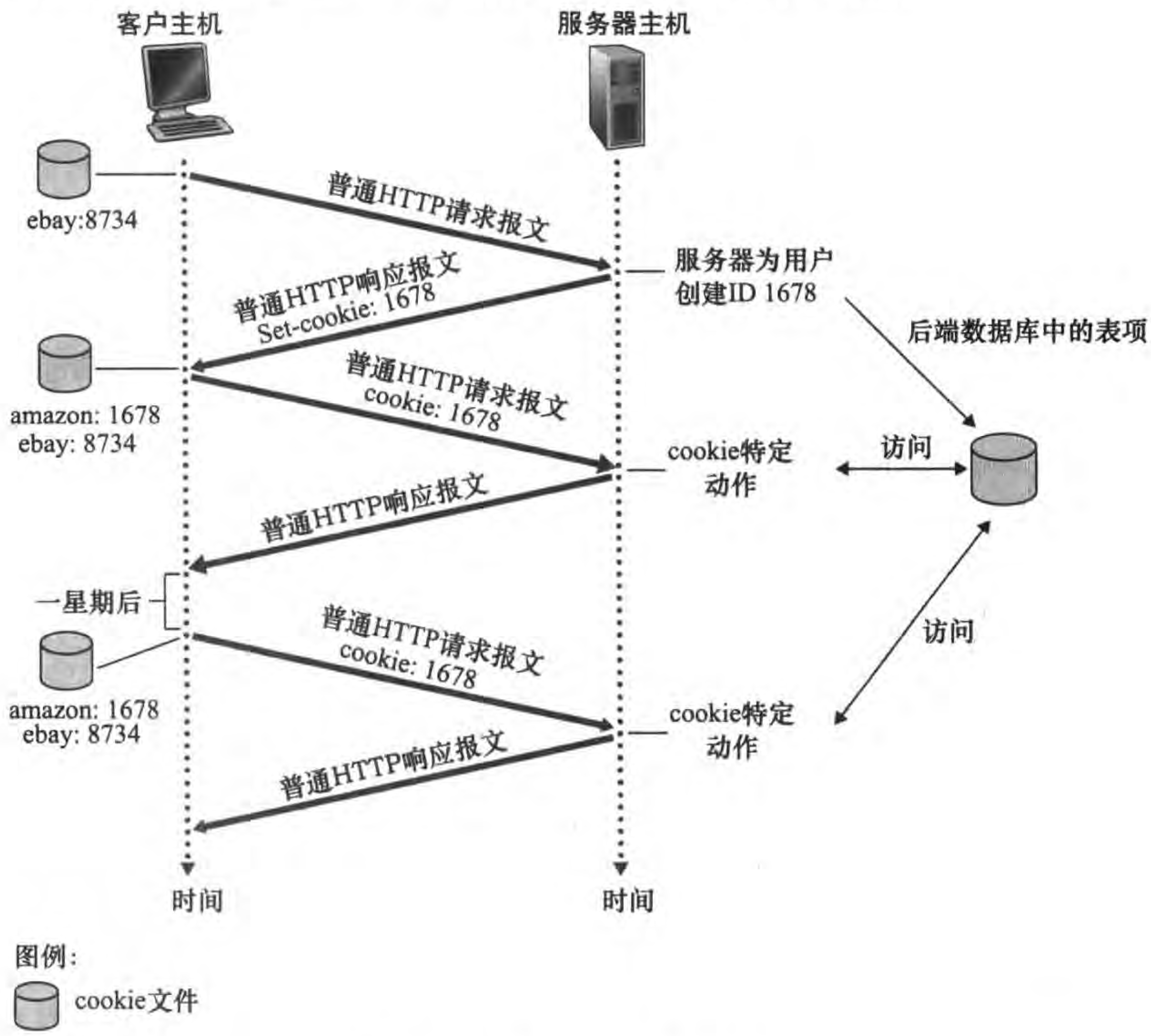


图 2-10 用 cookie 跟踪用户状态

如果 Susan 再次访问 Amazon 站点，比如说一个星期后，她的浏览器会在其请求报文中继续放入首部行 cookie：1678。Amazon 将根据 Susan 过去在 Amazon 访问的网页向她推荐产品。如果 Susan 也在 Amazon 注册过，即提供了她的全名、电子邮件地址、邮政地址和信用卡账号，则 Amazon 能在其数据库中包括这些信息，将 Susan 的名字与识别码相关联（以及她在过去访问过的本站点的所有页面）。这就解释了 Amazon 和其他一些电子商务网站实现"点击购物"（one-click shopping）的道理，即当 Susan 在后继的访问中选择购买某个物品时，她不必重新输入姓名、信用卡账号或者地址等信息了。

从上述讨论中我们看到，cookie 可以用于标识一个用户。用户首次访问一个站点时，可能需要提供一个用户标识（可能是名字）。在后继会话中，浏览器向服务器传递一个 cookie 首部，从而向该服务器标识了用户。因此 cookie 可以在无状态的 HTTP 之上建立一个用户会话层。例如，当用户向一个基于 Web 的电子邮件系统（如 Hotmail）注册时，浏览器向服务器发送 cookie 信息，允许该服务器在用户与应用程序会话的过程中标识该用户。

尽管 cookie 常常能简化用户的因特网购物活动，但是它的使用仍具有争议，因为它们被认为是对用户隐私的一种侵害。如我们刚才所见，结合 cookie 和用户提供的账户信息，Web 站点可以知道许多有关用户的信息，并可能将这些信息卖给第三方。Cookie Central [Cookie Central 2016] 包括了对 cookie 争论的广泛信息。

2.2.5 Web 缓存

Web 缓存器（Web cache）也叫**代理服务器**（proxy server），它是能够代表初始 Web 服务器来满足 HTTP 请求的网络实体。Web 缓存器有自己的磁盘存储空间，并在存储空间中保存最近请求过的对象的副本。如图 2-11 所示，可以配置用户的浏览器，使得用户的所有 HTTP 请求首先指向 Web 缓存器。一旦某浏览器被配置，每个对某对象的浏览器请求首先被定向到该 Web 缓存器。举例来说，假设浏览器正在请求对象 http://www.someschool.edu/campus.gif，将会发生如下情况：



图 2-11 客户通过 Web 缓存器请求对象

1）浏览器创建一个到 Web 缓存器的 TCP 连接，并向 Web 缓存器中的对象发送一个 HTTP 请求。

2）Web 缓存器进行检查，看看本地是否存储了该对象副本。如果有，Web 缓存器就向客户浏览器用 HTTP 响应报文返回该对象。

3）如果 Web 缓存器中没有该对象，它就打开一个与该对象的初始服务器（即 www.someschool.edu）的 TCP 连接。Web 缓存器则在这个缓存器到服务器的 TCP 连接上发送一个对该对象的 HTTP 请求。在收到该请求后，初始服务器向该 Web 缓存器发送具有该对象的 HTTP 响应。

4）当 Web 缓存器接收到该对象时，它在本地存储空间存储一份副本，并向客户的浏览器用 HTTP 响应报文发送该副本（通过现有的客户浏览器和 Web 缓存器之间的 TCP 连接）。

值得注意的是 Web 缓存器既是服务器又是客户。当它接收浏览器的请求并发回响应时，它是一个服务器。当它向初始服务器发出请求并接收响应时，它是一个客户。

Web 缓存器通常由 ISP 购买并安装。例如，一所大学可能在它的校园网上安装一台缓存器，并且将所有校园网上的用户浏览器配置为指向它。或者，一个主要的住宅 ISP（例如 Comcast）可能在它的网络上安装一台或多台 Web 缓存器，并且预先配置其配套的浏览器指向这些缓存器。

在因特网上部署 Web 缓存器有两个原因。首先，Web 缓存器可以大大减少对客户请求的响应时间，特别是当客户与初始服务器之间的瓶颈带宽远低于客户与 Web 缓存器之间的瓶颈带宽时更是如此。如果在客户与 Web 缓存器之间有一个高速连接（情况常常如此），并且如果用户所请求的对象在 Web 缓存器上，则 Web 缓存器可以迅速将该对象交付给用户。其次，如我们马上用例子说明的那样，Web 缓存器能够大大减少一个机构的接入链路到因特网的通信量。通过减少通信量，该机构（如一家公司或者一所大学）就不必急于增加带宽，因此降低了费用。此外，Web 缓存器能从整体上大大减低因特网上的 Web

流量，从而改善了所有应用的性能。

为了深刻理解缓存器带来的好处，我们考虑在图 2-12 场景下的一个例子。该图显示了两个网络，即机构（内部）网络和公共因特网的一部分。机构网络是一个高速的局域网，它的一台路由器与因特网上的一台路由器通过一条 15Mbps 的链路连接。这些初始服务器与因特网相连但位于全世界各地。假设对象的平均长度为 1Mb，从机构内的浏览器对这些初始服务器的平均访问速率为每秒 15 个请求。假设 HTTP 请求报文小到可以忽略，因而不会在网络中以及接入链路（从机构内部路由器到因特网路由器）上产生什么通信量。我们还假设在图 2-12 中从因特网接入链路一侧的路由器转发 HTTP 请求报文（在一个 IP 数据报中）开始，到它收到其响应报文（通常在多个 IP 数据报中）为止的时间平均为 2 秒。我们非正式地将该持续时延称为“因特网时延”。

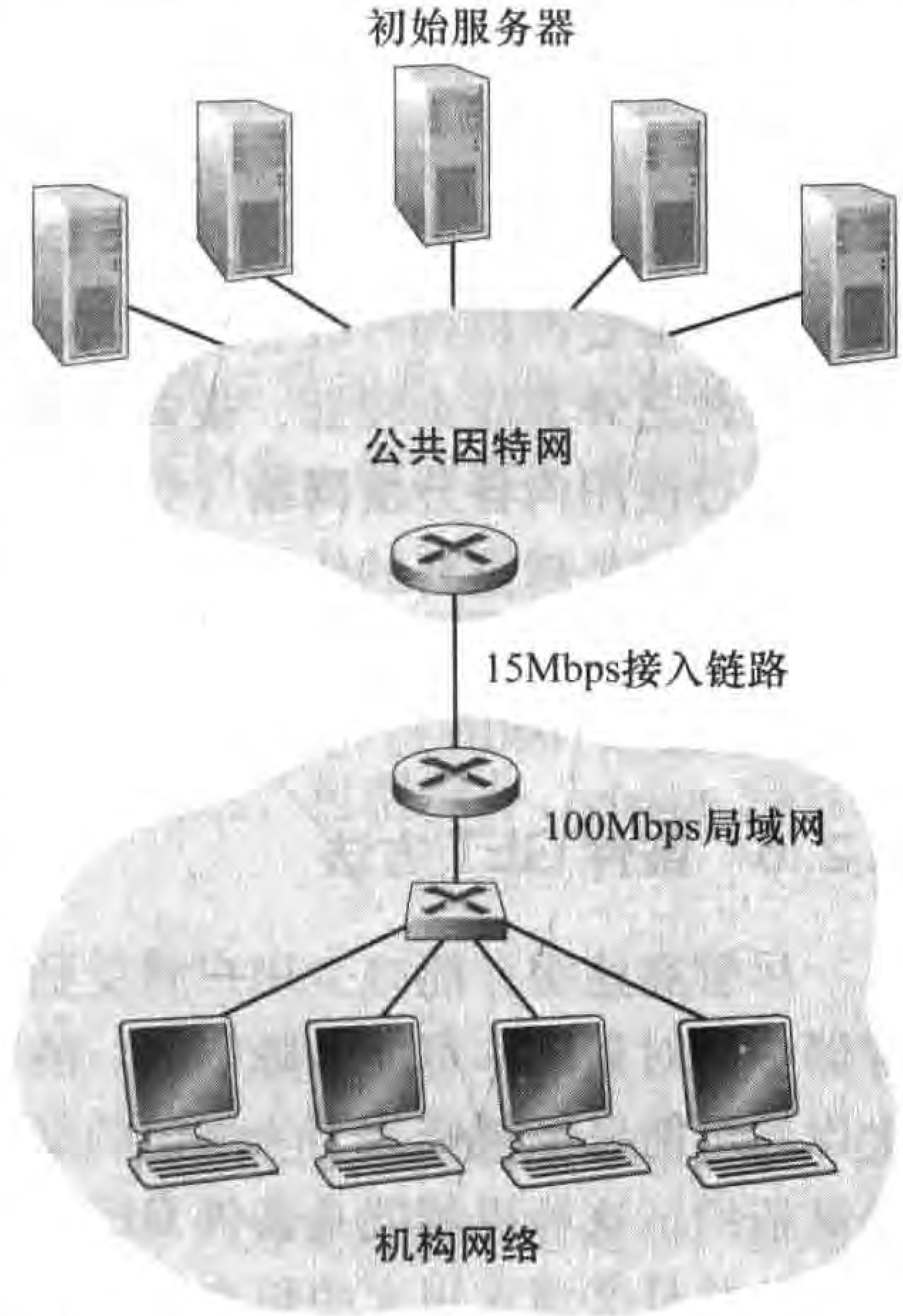


图 2-12　一个机构网络与因特网之间的瓶颈

总的响应时间，即从浏览器请求一个对象到接收到该对象为止的时间，是局域网时延、接入时延（即两台路由器之间的时延）和因特网时延之和。我们来粗略地估算一下这个时延。局域网上的流量强度（参见 1.4.2 节）为

(15 个请求/s) × (1Mb/请求)/(100Mbps) = 0.15

然而接入链路上的流量强度（从因特网路由器到机构路由器）为

(15 个请求/s) × (1Mb/请求)/(15Mbps) = 1

局域网上强度为 0.15 的通信量通常最多导致数十毫秒的时延，因此我们可以忽略局域网时延。然而，如在 1.4.2 节讨论的那样，如果流量强度接近 1（就像在图 2-12 中接入链路的情况那样），链路上的时延会变得非常大并且无限增长。因此，满足请求的平均响应时间将在分钟的量级上。显然，必须想办法来改进时间响应特性。

一个可能的解决办法就是增加接入链路的速率，如从 15Mbps 增加到 100Mbps。这可以将接入链路上的流量强度减少到 0.15，这样一来，两台路由器之间的链路时延也可以忽略了。这时，总的响应时间将大约为 2 秒，即为因特网时延。但这种解决方案也意味着该机构必须将它的接入链路由 15Mbps 升级为 100Mbps，这是一种代价很高的方案。

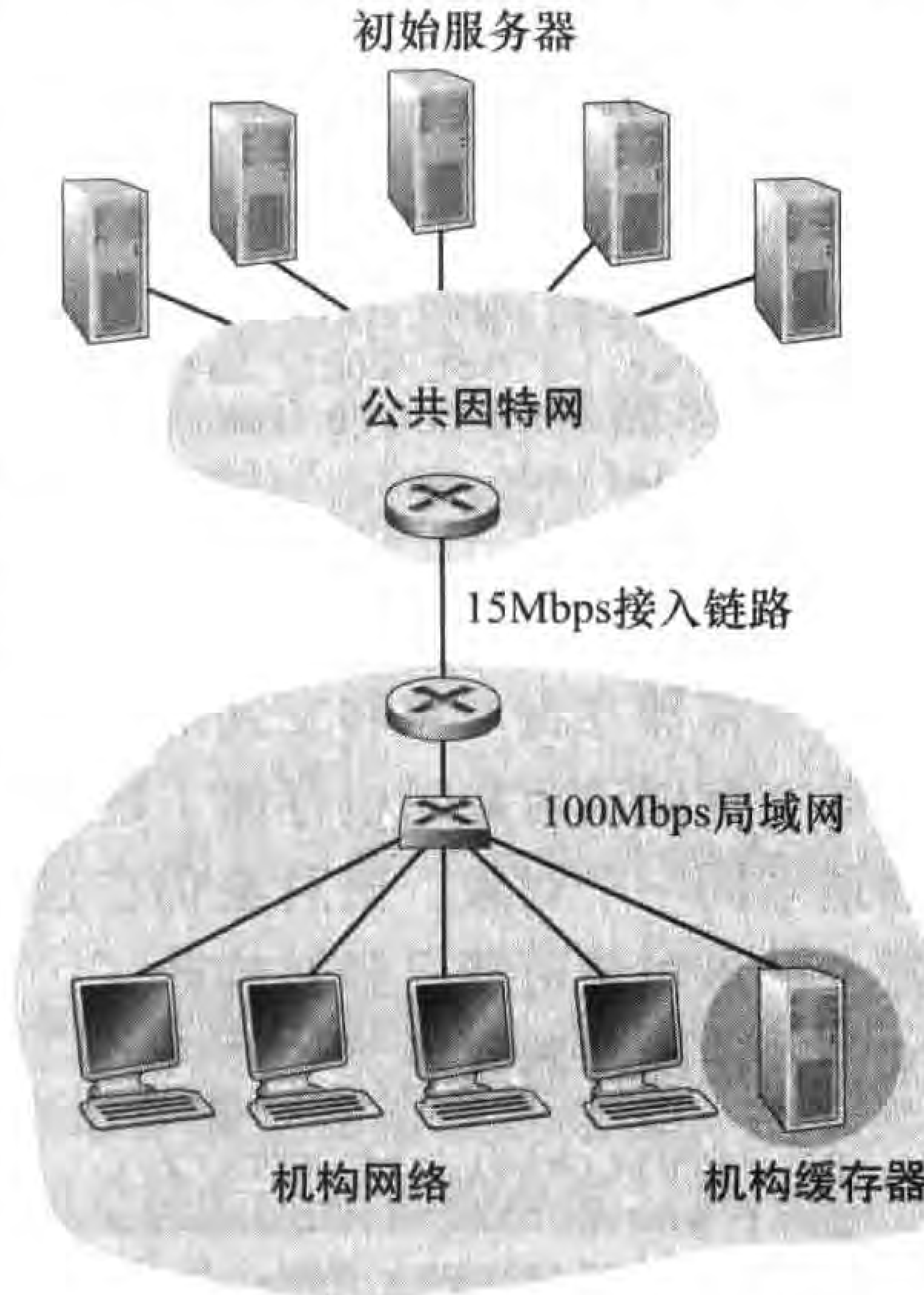


图 2-13　为机构网络添加一台缓存器

现在来考虑另一种解决方案，即不升级链路带宽而是在机构网络中安装一个 Web 缓存器。这种解决方案如图 2-13 所示。实践中的命中率（即由一个缓存器所满足的请求的比率）通常在 0.2 ~ 0.7 之间。为了便于阐述，我们假设该机构的缓存

命中率为0.4。因为客户和缓存连接在一个相同的高速局域网上，这样40%的请求将几乎立即会由缓存器得到响应，时延约在10ms以内。然而，剩下的60%的请求仍然要由初始服务器来满足。但是只有60%的被请求对象通过接入链路，在接入链路上的流量强度从1.0减小到0.6。一般而言，在15Mbps链路上，当流量强度小于0.8时对应的时延较小，约为几十毫秒。这个时延与2秒因特网时延相比是微不足道的。考虑这些之后，平均时延因此为

$$0.4\times(0.010\text{ 秒})+0.6\times(2.01\text{ 秒})$$

这略大于1.2秒。因此，第二种解决方案提供的响应时延甚至比第一种解决方案更低，也不需要该机构升级它到因特网的链路。该机构理所当然地要购买和安装Web缓存器。除此之外其成本较低，很多缓存器使用了运行在廉价PC上的公共域软件。

通过使用**内容分发网络**（Content Distribution Network，CDN），Web缓存器正在因特网中发挥着越来越重要的作用。CDN公司在因特网上安装了许多地理上分散的缓存器，因而使大量流量实现了本地化。有多个共享的CDN（例如Akamai和Limelight）和专用的CDN（例如谷歌和Netflix）。我们将在2.6节中更为详细地讨论CDN。

2.2.6 条件GET方法

尽管高速缓存能减少用户感受到的响应时间，但也引入了一个新的问题，即存放在缓存器中的对象副本可能是陈旧的。换句话说，保存在服务器中的对象自该副本缓存在客户上以后可能已经被修改了。幸运的是，HTTP协议有一种机制，允许缓存器证实它的对象是最新的。这种机制就是**条件GET**（conditional GET）方法。如果：①请求报文使用GET方法；并且②请求报文中包含一个“If-Modified-Since:”首部行。那么，这个HTTP请求报文就是一个条件GET请求报文。

为了说明GET方法的操作方式，我们看一个例子。首先，一个代理缓存器（proxy cache）代表一个请求浏览器，向某Web服务器发送一个请求报文：

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

其次，该Web服务器向缓存器发送具有被请求的对象的响应报文：

```
HTTP/1.1 200 OK
Date: Sat, 3 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 9 Sep 2015 09:23:24
Content-Type: image/gif

(data data data data data ...)
```

该缓存器在将对象转发到请求的浏览器的同时，也在本地缓存了该对象。重要的是，缓存器在存储该对象时也存储了最后修改日期。最后，一个星期后，另一个用户经过该缓存器请求同一个对象，该对象仍在这个缓存器中。由于在过去的一个星期中位于Web服务器上的该对象可能已经被修改了，该缓存器通过发送一个条件GET执行最新检查。具体来说，该缓存器发送：

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

值得注意的是If-Modified-Since：首部行的值正好等于一星期前服务器发送的响应报文

中的 Last-Modified：首部行的值。该条件 GET 报文告诉服务器，仅当自指定日期之后该对象被修改过，才发送该对象。假设该对象自 2015 年 9 月 9 日 09：23：24 后没有被修改。接下来的第四步，Web 服务器向该缓存器发送一个响应报文：

```
HTTP/1.1 304 Not Modified
Date: Sat, 10 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)

(empty entity body)
```

我们看到，作为对该条件 GET 方法的响应，该 Web 服务器仍发送一个响应报文，但并没有在该响应报文中包含所请求的对象。包含该对象只会浪费带宽，并增加用户感受到的响应时间，特别是如果该对象很大的时候更是如此。值得注意的是在最后的响应报文中，状态行中为 304 Not Modified，它告诉缓存器可以使用该对象，能向请求的浏览器转发它（该代理缓存器）缓存的该对象副本。

我们现在完成了对 HTTP 的讨论，这是我们详细学习的第一个因特网协议（应用层协议）。我们已经学习了 HTTP 报文的格式，学习了当发送和接收这些报文时 Web 客户和服务器所采取的动作。我们还学习了一点 Web 应用程序基础设施，包括缓存、cookie 和后端数据库，所有这些都以某种方式与 HTTP 协议有关。

2.3 因特网中的电子邮件

自从有了因特网，电子邮件就在因特网上流行起来。当因特网还在襁褓中时，电子邮件已经成为最为流行的应用程序［Segaller 1998］，年复一年，它变得越来越精细，越来越强大。它仍然是当今因特网上最重要和实用的应用程序之一。

与普通邮件一样，电子邮件是一种异步通信媒介，即当人们方便时就可以收发邮件，不必与他人的计划进行协调。与普通邮件相比，电子邮件更为快速并且易于分发，而且价格便宜。现代电子邮件具有许多强大的特性，包括具有附件、超链接、HTML 格式文本和图片的报文。

在本节中，我们将讨论位于因特网电子邮件的核心地位的应用层协议。在深入讨论这些应用层协议之前，我们先总体上看看因特网电子邮件系统和它的关键组件。

图 2-14 给出了因特网电子邮件系统的总体情况。从该图中我们可以看到它有 3 个主要组成部分：**用户代理**（user agent）、**邮件服务器**（mail server）和**简单邮件传输协议**（Simple Mail Transfer Protocol，SMTP）。下面我们结合发送方 Alice 发电子邮件给接收方 Bob 的场景，对每个组成部分进行描述。用户代理允许用户阅读、回复、转发、保存和撰写报文。微软的 Outlook 和 Apple Mail 是电子邮件用户代理的例子。当 Alice 完成邮件撰写时，她的邮件代理向其邮件服务器发送邮件，此时邮件放在邮件服务器的外出报文队列中。当 Bob 要阅读报文时，他的用户代理在其邮件服务器的邮箱中取得该报文。

邮件服务器形成了电子邮件体系结构的核心。每个接收方（如 Bob）在其中的某个邮件服务器上有一个**邮箱**（mailbox）。Bob 的邮箱管理和维护着发送给他的报文。一个典型的邮件发送过程是：从发送方的用户代理开始，传输到发送方的邮件服务器，再传输到接收方的邮件服务器，然后在这里被分发到接收方的邮箱中。当 Bob 要在他的邮箱中读取该报文时，包含他邮箱的邮件服务器（使用用户名和口令）来鉴别 Bob。Alice 的邮箱也必

须能处理Bob的邮件服务器的故障。如果Alice的服务器不能将邮件交付给Bob的服务器，Alice的邮件服务器在一个**报文队列**（message queue）中保持该报文并在以后尝试再次发送。通常每30分钟左右进行一次尝试；如果几天后仍不能成功，服务器就删除该报文并以电子邮件的形式通知发送方（Alice）。

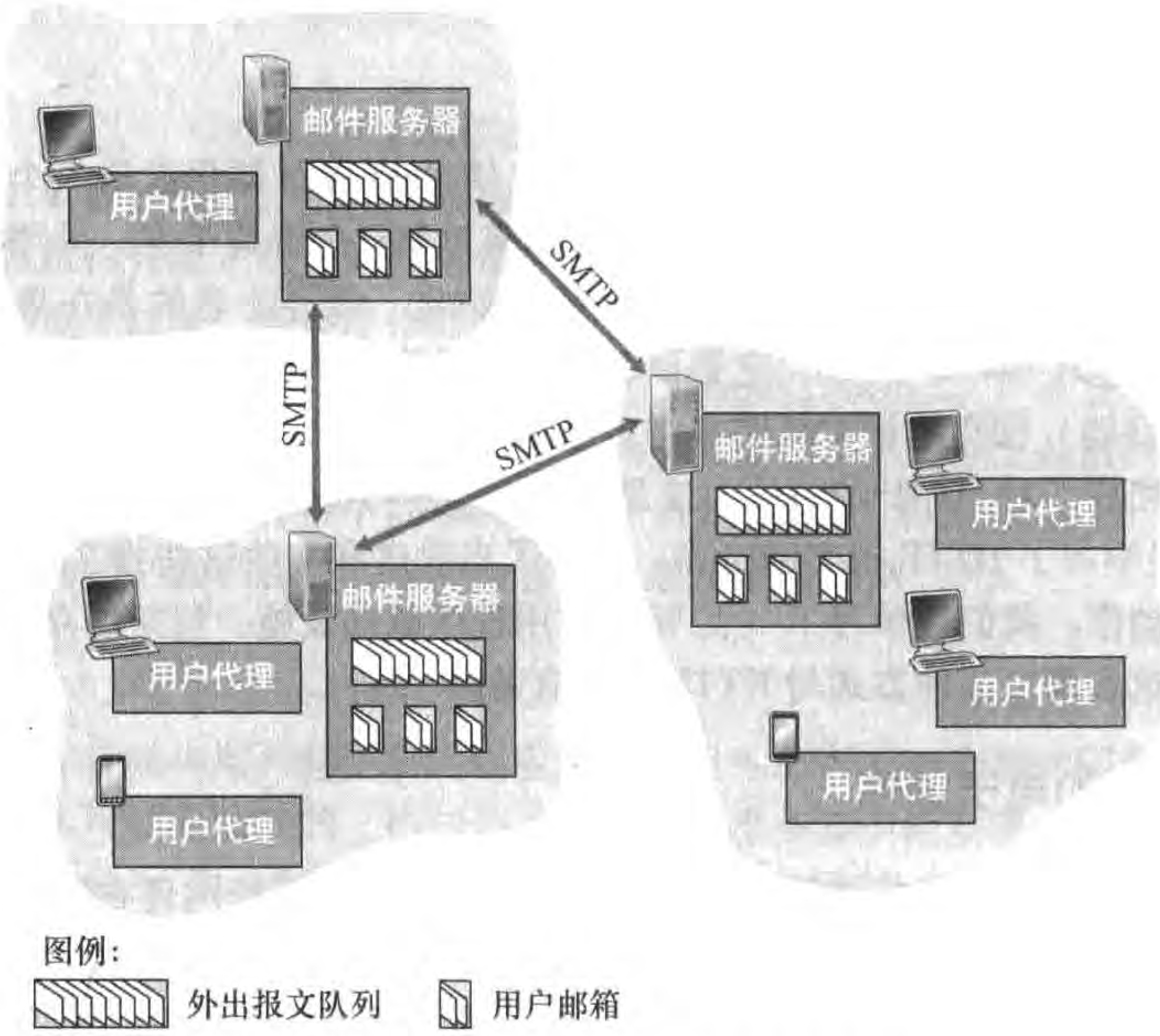


图2-14　因特网电子邮件系统的总体描述

SMTP是因特网电子邮件中主要的应用层协议。它使用TCP可靠数据传输服务，从发送方的邮件服务器向接收方的邮件服务器发送邮件。像大多数应用层协议一样，SMTP也有两个部分：运行在发送方邮件服务器的客户端和运行在接收方邮件服务器的服务器端。每台邮件服务器上既运行SMTP的客户端也运行SMTP的服务器端。当一个邮件服务器向其他邮件服务器发送邮件时，它就表现为SMTP的客户；当邮件服务器从其他邮件服务器上接收邮件时，它就表现为一个SMTP的服务器。

2.3.1　SMTP

RFC 5321给出了SMTP的定义。SMTP是因特网电子邮件的核心。如前所述，SMTP用于从发送方的邮件服务器发送报文到接收方的邮件服务器。SMTP问世的时间比HTTP要长得多（初始的SMTP协议的RFC可追溯到1982年，而SMTP在此之前很长一段时间就已经出现了）。尽管电子邮件应用在因特网上的独特地位可以证明SMTP有着众多非常出色的性质，但它所具有的某种陈旧特征表明它仍然是一种继承的技术。例如，它限制所有邮件报文的体部分（不只是其首部）只能采用简单的7比特ASCII表示。在20世纪80年代早期，这种限制是明智的，因为当时传输能力不足，没有人会通过电子邮件发送大的附件或是大的图片、声音或者视频文件。然而，在今天的多媒体时代，7位ASCII的限制的确有点痛苦，即在用SMTP传送邮件之前，需要将二进制多媒体数据编码为ASCII码，并且在使用SMTP传输后要求将相应的

ASCII 码邮件解码还原为多媒体数据。2.2 节讲过，使用 HTTP 传送前不需要将多媒体数据编码为 ASCII 码。

为了描述 SMTP 的基本操作，我们观察一种常见的情景。假设 Alice 想给 Bob 发送一封简单的 ASCII 报文。

1）Alice 调用她的邮件代理程序并提供 Bob 的邮件地址（例如 bob@someschool.edu），撰写报文，然后指示用户代理发送该报文。

2）Alice 的用户代理把报文发给她的邮件服务器，在那里该报文被放在报文队列中。

3）运行在 Alice 的邮件服务器上的 SMTP 客户端发现了报文队列中的这个报文，它就创建一个到运行在 Bob 的邮件服务器上的 SMTP 服务器的 TCP 连接。

4）在经过一些初始 SMTP 握手后，SMTP 客户通过该 TCP 连接发送 Alice 的报文。

5）在 Bob 的邮件服务器上，SMTP 的服务器端接收该报文。Bob 的邮件服务器然后将该报文放入 Bob 的邮箱中。

6）在 Bob 方便的时候，他调用用户代理阅读该报文。

图 2-15 总结了上述这个情况。

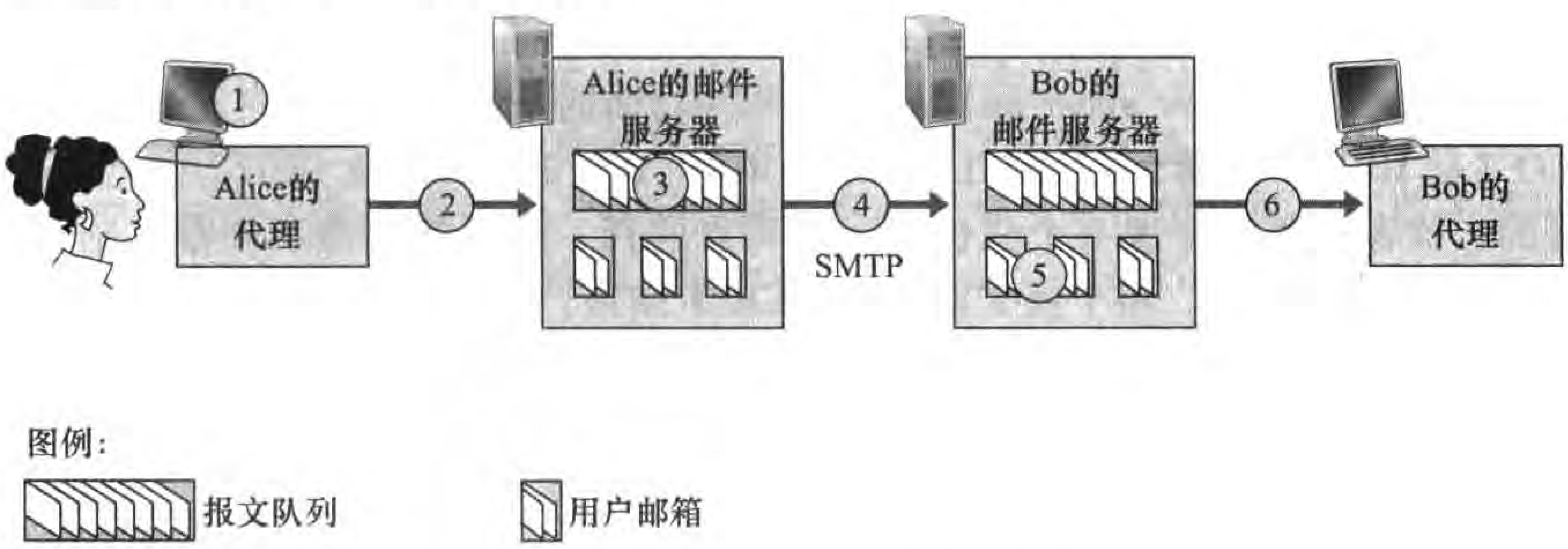


图 2-15 Alice 向 Bob 发送一条报文

观察到下述现象是重要的：SMTP 一般不使用中间邮件服务器发送邮件，即使这两个邮件服务器位于地球的两端也是这样。假设 Alice 的邮件服务器在中国香港，而 Bob 的服务器在美国圣路易斯，那么这个 TCP 连接也是从香港服务器到圣路易斯服务器之间的直接相连。特别是，如果 Bob 的邮件服务器没有开机，该报文会保留在 Alice 的邮件服务器上并等待进行新的尝试，这意味着邮件并不在中间的某个邮件服务器存留。

我们现在仔细观察一下，SMTP 是如何将一个报文从发送邮件服务器传送到接收邮件服务器的。我们将看到，SMTP 与人类面对面交往的行为方式有许多类似性。首先，客户 SMTP（运行在发送邮件服务器主机上）在 25 号端口建立一个到服务器 SMTP（运行在接收邮件服务器主机上）的 TCP 连接。如果服务器没有开机，客户会在稍后继续尝试连接。一旦连接建立，服务器和客户执行某些应用层的握手，就像人们在互相交流前先进行自我介绍一样。SMTP 的客户和服务器在传输信息前先相互介绍。在 SMTP 握手的阶段，SMTP 客户指示发送方的邮件地址（产生报文的那个人）和接收方的邮件地址。一旦该 SMTP 客户和服务器彼此介绍之后，客户发送该报文。SMTP 能依赖 TCP 提供的可靠数据传输无差错地将邮件投递到接收服务器。该客户如果有另外的报文要发送到该服务器，就在该相同的 TCP 连接上重复这种处理；否则，它指示 TCP 关闭连接。

接下来我们分析一个在 SMTP 客户（C）和 SMTP 服务器（S）之间交换报文文本的例子。客户的主机名为 crepes. fr，服务器的主机名为 hamburger. edu。以 C：开头的 ASCII 码文本行正是客户交给其 TCP 套接字的那些行，以 S：开头的 ASCII 码则是服务器发送给其 TCP 套接字的那些行。一旦创建了 TCP 连接，就开始了下列过程。

```
S:  220 hamburger.edu
C:  HELO crepes.fr
S:  250 Hello crepes.fr, pleased to meet you
C:  MAIL FROM: <alice@crepes.fr>
S:  250 alice@crepes.fr ... Sender ok
C:  RCPT TO: <bob@hamburger.edu>
S:  250 bob@hamburger.edu ... Recipient ok
C:  DATA
S:  354 Enter mail, end with "." on a line by itself
C:  Do you like ketchup?
C:  How about pickles?
C:  .
S:  250 Message accepted for delivery
C:  QUIT
S:  221 hamburger.edu closing connection
```

在上例中，客户从邮件服务器 crepes. fr 向邮件服务器 hamburger. edu 发送了一个报文（“Do you like ketchup? How about pickles?”）。作为对话的一部分，该客户发送了 5 条命令：HELO（是 HELLO 的缩写）、MAIL FROM、RCPT TO、DATA 以及 QUIT。这些命令都是自解释的。该客户通过发送一个只包含一个句点的行，向服务器指示该报文结束了。（按照 ASCII 码的表示方法，每个报文以 CRLF. CRLF 结束，其中的 CR 和 LF 分别表示回车和换行。）服务器对每条命令做出回答，其中每个回答含有一个回答码和一些（可选的）英文解释。我们在这里指出 SMTP 用的是持续连接：如果发送邮件服务器有几个报文发往同一个接收邮件服务器，它可以通过同一个 TCP 连接发送这些所有的报文。对每个报文，该客户用一个新的 MAIL FROM：crepes. fr 开始，用一个独立的句点指示该邮件的结束，并且仅当所有邮件发送完后才发送 QUIT。

我们强烈推荐你使用 Telnet 与一个 SMTP 服务器进行一次直接对话。使用的命令是

```
telnet serverName 25
```

其中 serverName 是本地邮件服务器的名称。当你这么做时，就直接在本地主机与邮件服务器之间建立一个 TCP 连接。输完上述命令后，你立即会从该服务器收到 220 回答。接下来，在适当的时机发出 HELO、MAIL FROM、RCPT TO、DATA、CRLF. CRLF 以及 QUIT 等 SMTP 命令。强烈推荐你做本章后面的编程作业 3。在该作业中，你将在 SMTP 的客户端实现一个简单的用户代理，它允许你经本地邮件服务器向任意的接收方发送电子邮件报文。

2.3.2　与 HTTP 的对比

我们简要地比较一下 SMTP 和 HTTP。这两个协议都用于从一台主机向另一台主机传送文件：HTTP 从 Web 服务器向 Web 客户（通常是一个浏览器）传送文件（也称为对象）；SMTP 从一个邮件服务器向另一个邮件服务器传送文件（即电子邮件报文）。当进行文件传送时，持续的 HTTP 和 SMTP 都使用持续连接。因此，这两个协议有一些共同特征。然而，两者之间也有一些重要的区别。首先，HTTP 主要是一个**拉协议**（pull protocol），即在方便的时候，某些人在 Web 服务器上装载信息，用户使用 HTTP 从该服务器拉取这些

信息。特别是 TCP 连接是由想接收文件的机器发起的。另一方面，SMTP 基本上是一个**推协议**（push protocol），即发送邮件服务器把文件推向接收邮件服务器。特别是，这个 TCP 连接是由要发送该文件的机器发起的。

第二个区别就是我们前面间接地提到过的，SMTP 要求每个报文（包括它们的体）采用7 比特 ASCII 码格式。如果某报文包含了非 7 比特 ASCII 字符（如具有重音的法文字符）或二进制数据（如图形文件），则该报文必须按照7 比特 ASCII 码进行编码。HTTP 数据则不受这种限制。

第三个重要区别是如何处理一个既包含文本又包含图形（也可能是其他媒体类型）的文档。如我们在 2.2 节知道的那样，HTTP 把每个对象封装到它自己的 HTTP 响应报文中，而 SMTP 则把所有报文对象放在一个报文之中。

2.3.3 邮件报文格式

当 Alice 给 Bob 写一封邮寄时间很长的普通信件时，她可能要在信的上部包含各种各样的环境首部信息，如 Bob 的地址、她自己的回复地址以及日期等。同样，当一个人给另一个人发送电子邮件时，一个包含环境信息的首部位于报文体前面。这些环境信息包括在一系列首部行中，这些行由 RFC 5322 定义。首部行和该报文的体用空行（即回车换行）进行分隔。RFC 5322 定义了邮件首部行和它们的语义解释的精确格式。如同 HTTP 协议，每个首部行包含了可读的文本，是由关键词后跟冒号及其值组成的。某些关键词是必需的，另一些则是可选的。每个首部必须含有一个 From：首部行和一个 To：首部行；一个首部也许包含一个 Subject：首部行以及其他可选的首部行。重要的是注意到下列事实：这些首部行不同于我们在 2.3.1 节所学到的 SMTP 命令（即使那里包含了某些相同的词汇，如 from 和 to）。那节中的命令是 SMTP 握手协议的一部分；本节中考察的首部行则是邮件报文自身的一部分。

一个典型的报文首部看起来如下：

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

在报文首部之后，紧接着一个空白行，然后是以 ACSII 格式表示的报文体。你应当用 Telnet 向邮件服务器发送包含一些首部行的报文，包括 Subject：首部行。为此，输入命令 telnet serverName 25，如在 2.3.1 节中讨论的那样。

2.3.4 邮件访问协议

一旦 SMTP 将邮件报文从 Alice 的邮件服务器交付给 Bob 的邮件服务器，该报文就被放入了 Bob 的邮箱中。在此讨论中，我们按惯例假定 Bob 是通过登录到服务器主机，并直接在该主机上运行一个邮件阅读程序来阅读他的邮件的。直到 20 世纪 90 年代早期，这都是一种标准方式。而在今天，邮件访问使用了一种客户 - 服务器体系结构，即典型的用户通过在用户端系统上运行的客户程序来阅读电子邮件，这里的端系统可能是办公室的 PC、便携机或者是智能手机。通过在本地主机上运行邮件客户程序，用户享受一系列丰富的特性，包括查看多媒体报文和附件的能力。

假设 Bob（接收方）在其本地 PC 上运行用户代理程序，考虑在他的本地 PC 上也放置一个邮件服务器是自然而然的事。在这种情况下，Alice 的邮件服务器就能直接

与 Bob 的 PC 进行对话了。然而这种方法会有一个问题。前面讲过邮件服务器管理用户的邮箱，并且运行 SMTP 的客户端和服务器端。如果 Bob 的邮件服务器位于他的 PC 上，那么为了能够及时接收可能在任何时候到达的新邮件，他的 PC 必须总是不间断地运行着并一直保持在线。这对于许多因特网用户而言是不现实的。相反，典型的用户通常在本地 PC 上运行一个用户代理程序，而它访问存储在总是保持开机的共享邮件服务器上的邮箱。该邮件服务器与其他用户共享，并且通常由用户的 ISP 进行维护（如大学或公司）。

现在我们考虑当从 Alice 向 Bob 发送一个电子邮件报文时所取的路径。我们刚才已经知道，在沿着该路径的某些点上，需要将电子邮件报文存放在 Bob 的邮件服务器上。通过让Alice 的用户代理直接向 Bob 的邮件服务器发送报文，就能够做到这一点。这能够由 SMTP 来完成：实际上，SMTP 被设计成将电子邮件从一台主机推到另一台主机。然而，通常Alice 的用户代理和 Bob 的邮件服务器之间并没有一个直接的 SMTP 对话。相反，如图 2-16 所示，Alice 的用户代理用 SMTP 将电子邮件报文推入她的邮件服务器，接着她的邮件服务器（作为一个 SMTP 客户）再用 SMTP 将该邮件中继到 Bob 的邮件服务器。为什么该过程要分成两步呢？主要是因为不通过 Alice 的邮件服务器进行中继，Alice 的用户代理将没有任何办法到达一个不可达的目的地接收服务器。通过首先将邮件存放在自己的邮件服务器中，Alice 的邮件服务器可以重复地尝试向 Bob 的邮件服务器发送该报文，如每 30 分钟一次，直到 Bob 的邮件服务器变得运行为止。（并且如果 Alice 的邮件服务器关机，她则能向系统管理员进行申告！）SMTP RFC 文档定义了如何使用 SMTP 命令经过多个 SMTP 服务器进行报文中继。

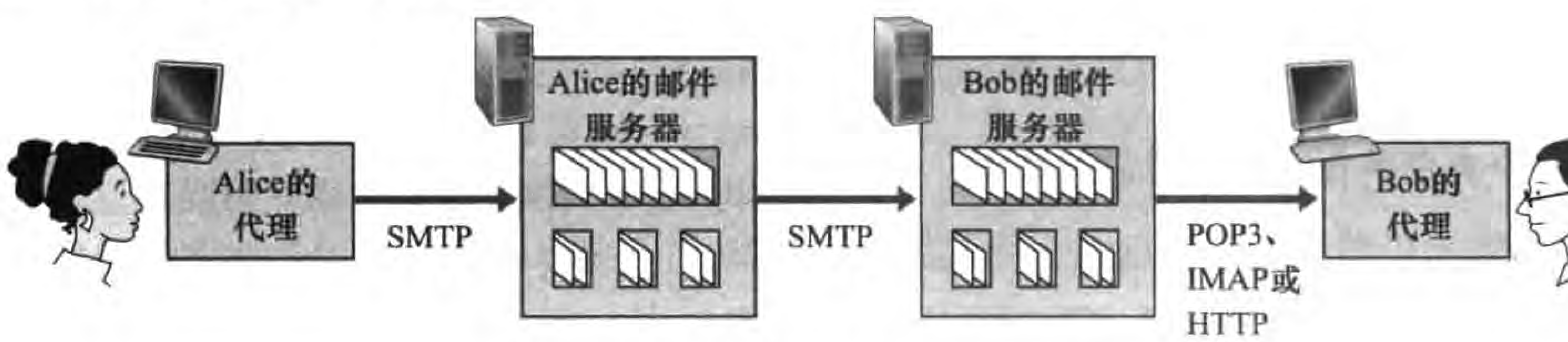


图 2-16 电子邮件协议及其通信实体

但是对于该难题仍然有一个疏漏的环节！像 Bob 这样的接收方，是如何通过运行其本地 PC 上的用户代理，获得位于他的某 ISP 的邮件服务器上的邮件呢？值得注意的是 Bob 的用户代理不能使用 SMTP 得到报文，因为取报文是一个拉操作，而 SMTP 协议是一个推协议。通过引入一个特殊的邮件访问协议来解决这个难题，该协议将 Bob 邮件服务器上的报文传送给他的本地 PC。目前有一些流行的邮件访问协议，包括**第三版的邮局协议**（Post Office Protocol—Version 3，POP3）、**因特网邮件访问协议**（Internet Mail Access Protocol，IMAP）以及 HTTP。

图 2-16 总结了应用于因特网电子邮件的一些协议：SMTP 用来将邮件从发送方的邮件服务器传输到接收方的邮件服务器；SMTP 也用来将邮件从发送方的用户代理传送到发送方的邮件服务器。如 POP3 这样的邮件访问协议用来将邮件从接收方的邮件服务器传送到接收方的用户代理。

1. POP3

POP3 是一个极为简单的邮件访问协议，由 RFC 1939 进行定义。文档 RFC 1939 简短且可读性强。因为该协议非常简单，故其功能相当有限。当用户代理（客户）打开了一个到邮件服务器（服务器）端口 110 上的 TCP 连接后，POP3 就开始工作了。随着建立 TCP 连接，POP3 按照三个阶段进行工作：特许（authorization）、事务处理以及更新。在第一个阶段即特许阶段，用户代理发送（以明文形式）用户名和口令以鉴别用户。在第二个阶段即事务处理阶段，用户代理取回报文；同时在这个阶段用户代理还能进行如下操作，对报文做删除标记，取消报文删除标记，以及获取邮件的统计信息。在第三个阶段即更新阶段，它出现在客户发出了 quit 命令之后，目的是结束该 POP3 会话；这时，该邮件服务器删除那些被标记为删除的报文。

在 POP3 的事务处理过程中，用户代理发出一些命令，服务器对每个命令做出回答。回答可能有两种：+OK（有时后面还跟有服务器到客户的数据），被服务器用来指示前面的命令是正常的；-ERR，被服务器用来指示前面的命令出现了某些差错。

特许阶段有两个主要的命令：user <user name> 和 pass <password>。为了举例说明这两个命令，我们建议你直接用 Telnet 登录到 POP3 服务器的 110 端口，然后发出这两个命令。假设邮件服务器的名字为 mailServer，那么你将看到类似的过程：

```
telnet mailServer 110
+OK POP3 server ready
user bob
+OK
pass hungry
+OK user successfully logged on
```

如果你的命令拼写错了，该 POP3 服务器将返回一个 -ERR 报文。

现在我们来看一下事务处理过程。使用 POP3 的用户代理通常被用户配置为“下载并删除”或者“下载并保留”方式。POP3 用户代理发出的命令序列取决于用户代理程序被配置为这两种工作方式的哪一种。使用下载并删除方式，用户代理发出 list、retr 和 dele 命令。举例来说，假设用户在他（她）的邮箱里有两个报文。在下面的对话中，C：（代表客户）是用户代理，S：（代表服务器）是邮件服务器。事务处理过程将类似于如下过程：

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: (blah blah ...
S: .................
S: ..........blah)
S: .
C: dele 1
C: retr 2
S: (blah blah ...
S: .................
S: ..........blah)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

用户代理首先请求邮件服务器列出所有存储的报文的长度。接着用户代理从邮件服务

器取回并删除每封邮件。注意到在特许阶段以后，用户代理仅使用四个命令 list、retr、dele 和 quit，这些命令的语法定义在 RFC 1939 中。在处理 quit 命令后，POP3 服务器进入更新阶段，从用户的邮箱中删除邮件 1 和 2。

使用下载并删除方式存在的问题是，邮件接收方 Bob 可能是移动的，可能希望从多个不同的机器访问他的邮件报文，如从办公室的 PC、家里的 PC 或他的便携机来访问邮件。下载并删除方式将对 Bob 的邮件报文根据这 3 台机器进行划分，特别是如果 Bob 最先是在他办公室的 PC 上收取了一条邮件，那么晚上当他在家里时，通过他的便携机将不能再收取该邮件。使用下载并保留方式，用户代理下载某邮件后，该邮件仍保留在邮件服务器上。这时，Bob 就能通过不同的机器重新读取这些邮件；他能在工作时收取一封报文，而在工作回家后再次访问它。

在用户代理与邮件服务器之间的 POP3 会话期间，该 POP3 服务器保留了一些状态信息；特别是记录了哪些用户报文被标记为删除了。然而，POP3 服务器并不在 POP3 会话过程中携带状态信息。会话中不包括状态信息大大简化了 POP3 服务的实现。

2. IMAP

使用 POP3 访问时，一旦 Bob 将邮件下载到本地主机后，他就能建立邮件文件夹，并将下载的邮件放入该文件夹中。然后 Bob 可以删除报文，在文件夹之间移动报文，并查询报文（通过发送方的名字或报文主题）。但是这种文件夹和报文存放在本地主机上的方式，会给移动用户带来问题，因为他更喜欢使用一个在远程服务器上的层次文件夹，这样他可以从任何一台机器上对所有报文进行访问。使用 POP3 是不可能做到这一点的，POP3 协议没有给用户提供任何创建远程文件夹并为报文指派文件夹的方法。

为了解决这个或其他一些问题，由 RFC 3501 定义的因特网邮件访问协议（IMAP）应运而生。和 POP3 一样，IMAP 是一个邮件访问协议，但是它比 POP3 具有更多的特色，不过也比 POP3 复杂得多。（因此客户和服务器端的实现也都复杂得多。）

IMAP 服务器把每个报文与一个文件夹联系起来；当报文第一次到达服务器时，它与收件人的 INBOX 文件夹相关联。收件人则能够把邮件移到一个新的、用户创建的文件夹中，阅读邮件，删除邮件等。IMAP 协议为用户提供了创建文件夹以及将邮件从一个文件夹移动到另一个文件夹的命令。IMAP 还为用户提供了在远程文件夹中查询邮件的命令，按指定条件去查询匹配的邮件。值得注意的是，与 POP3 不同，IMAP 服务器维护了 IMAP 会话的用户状态信息，例如，文件夹的名字以及哪些报文与哪些文件夹相关联。

IMAP 的另一个重要特性是它具有允许用户代理获取报文某些部分的命令。例如，一个用户代理可以只读取一个报文的报文首部，或只是一个多部分 MIME 报文的一部分。当用户代理和其邮件服务器之间使用低带宽连接（如一个低速调制解调器链路）的时候，这个特性非常有用。使用这种低带宽连接时，用户可能并不想取回他邮箱中的所有邮件，尤其要避免可能包含如音频或视频片断的大邮件。

3. 基于 Web 的电子邮件

今天越来越多的用户使用他们的 Web 浏览器收发电子邮件。20 世纪 90 年代中期 Hotmail 引入了基于 Web 的接入。今天，谷歌、雅虎以及几乎所有重要的大学或者公

司也提供了基于 Web 的电子邮件。使用这种服务，用户代理就是普通的浏览器，用户和他远程邮箱之间的通信则通过 HTTP 进行。当一个收件人（如 Bob），想从他的邮箱中访问一个报文时，该电子邮件报文从 Bob 的邮件服务器发送到他的浏览器，使用的是 HTTP 而不是 POP3 或者 IMAP 协议。当发件人（如 Alice）要发送一封电子邮件报文时，该电子邮件报文从 Alice 的浏览器发送到她的邮件服务器，使用的是 HTTP 而不是 SMTP。然而，Alice 的邮件服务器在与其他的邮件服务器之间发送和接收邮件时，仍然使用的是 SMTP。

2.4 DNS：因特网的目录服务

人类能以很多方式来标识。例如，我们能够通过出生证书上的名字来标识；能够通过社会保险号码来标识；也能够通过驾驶执照上的号码来标识。尽管这些标识办法都可以用来识别一个人，但是在特定环境下，某种识别方法可能比另一种方法更为适合。例如，IRS（美国那个声名狼藉的税务征收机构）的计算机更喜欢使用定长的社会保险号码而不是出生证书上的姓名。另一方面，普通人乐于使用更好记的出生证书上的姓名而不是社会保险号码。（毫无疑问，你能想象人们之间以这种方式说话吗？如“你好，我叫132-67-9875。请找一下我的丈夫 178-87-1146”。）

因特网上的主机和人类一样，可以使用多种方式进行标识。主机的一种标识方法是用它的**主机名**（hostname），如 www. facebook. com、www. google. com、gaia. cs. umass. edu 以及 cis. poly. edu 等，这些名字便于记忆也乐于被人们接受。然而，主机名几乎没有提供（即使有也很少）关于主机在因特网中位置的信息。（一个名为 www. eurecom. fr 的主机以国家码 . fr 结束，告诉我们该主机很可能在法国，仅此而已。）况且，因为主机名可能由不定长的字母数字组成，路由器难以处理。由于这些原因，主机也可以使用所谓 **IP 地址**（IP address）进行标识。

我们将在第 4 章更为详细地讨论 IP 地址，但现在简略地介绍一下还是有必要的。一个 IP 地址由 4 个字节组成，并有着严格的层次结构。例如 121. 7. 106. 83 这样一个 IP 地址，其中的每个字节都被句点分隔开来，表示了 0 ~ 255 的十进制数字。我们说 IP 地址具有层次结构，是因为当我们从左至右扫描它时，我们会得到越来越具体的关于主机位于因特网何处的信息（即在众多网络的哪个网络里）。类似地，当我们从下向上查看邮政地址时，我们能够获得该地址位于何处的越来越具体的信息。

2.4.1 DNS 提供的服务

我们刚刚看到了识别主机有两种方式，通过主机名或者 IP 地址。人们喜欢便于记忆的主机名标识方式，而路由器则喜欢定长的、有着层次结构的 IP 地址。为了折中这些不同的偏好，我们需要一种能进行主机名到 IP 地址转换的目录服务。这就是**域名系统**（Domain Name System，DNS）的主要任务。DNS 是：①一个由分层的 **DNS 服务器**（DNS server）实现的分布式数据库；②一个使得主机能够查询分布式数据库的应用层协议。DNS 服务器通常是运行 BIND（Berkeley Internet Name Domain）软件［BIND 2012］的 UNIX 机器。DNS 协议运行在 UDP 之上，使用 53 号端口。

实践原则

DNS：通过客户－服务器模式提供的重要网络功能

与HTTP、FTP和SMTP协议一样，DNS协议是应用层协议，其原因在于：①使用客户－服务器模式运行在通信的端系统之间；②在通信的端系统之间通过下面的端到端运输协议来传送DNS报文。然而，在其他意义上，DNS的作用非常不同于Web应用、文件传输应用以及电子邮件应用。与这些应用程序不同之处在于，DNS不是一个直接和用户打交道的应用。相反，DNS是为因特网上的用户应用程序以及其他软件提供一种核心功能，即将主机名转换为其背后的IP地址。我们在1.2节就提到，因特网体系结构的复杂性大多数位于网络的“边缘”。DNS通过采用了位于网络边缘的客户和服务器，实现了关键的名字到地址转换功能，它还是这种设计原理的另一个范例。

DNS通常是由其他应用层协议所使用的，包括HTTP、SMTP和FTP，将用户提供的主机名解析为IP地址。举一个例子，考虑运行在某用户主机上的一个浏览器（即一个HTTP客户）请求URL www.someschool.edu/index.html页面时会发生什么现象。为了使用户的主机能够将一个HTTP请求报文发送到Web服务器www.someschool.edu，该用户主机必须获得www.someschool.edu的IP地址。其做法如下。

1）同一台用户主机上运行着DNS应用的客户端。

2）浏览器从上述URL中抽取出主机名www.someschool.edu，并将这台主机名传给DNS应用的客户端。

3）DNS客户向DNS服务器发送一个包含主机名的请求。

4）DNS客户最终会收到一份回答报文，其中含有对应于该主机名的IP地址。

5）一旦浏览器接收到来自DNS的该IP地址，它能够向位于该IP地址80端口的HTTP服务器进程发起一个TCP连接。

从这个例子中，我们可以看到DNS给使用它的因特网应用带来了额外的时延，有时还相当可观。幸运的是，如我们下面讨论的那样，想获得的IP地址通常就缓存在一个“附近的”DNS服务器中，这有助于减少DNS的网络流量和DNS的平均时延。

除了进行主机名到IP地址的转换外，DNS还提供了一些重要的服务：

- **主机别名**（host aliasing）。有着复杂主机名的主机能拥有一个或者多个别名。例如，一台名为relay1.west-coast.enterprise.com的主机，可能还有两个别名为enterprise.com和www.enterprise.com。在这种情况下，relay1.west-coast.enterprise.com也称为**规范主机名**（canonical hostname）。主机别名（当存在时）比主机规范名更加容易记忆。应用程序可以调用DNS来获得主机别名对应的规范主机名以及主机的IP地址。
- **邮件服务器别名**（mail server aliasing）。显而易见，人们也非常希望电子邮件地址好记忆。例如，如果Bob在雅虎邮件上有一个账户，Bob的邮件地址就像bob@yahoo.com这样简单。然而，雅虎邮件服务器的主机名可能更为复杂，不像yahoo.com那样简单好记（例如，规范主机名可能像relay1.west-coast.hotmail.com那样）。电子邮件应用程序可以调用DNS，对提供的主机名别名进行解析，以获得该主机的规范主机名及其IP地址。事实上，MX记录（参见后面）允许一个公司

的邮件服务器和 Web 服务器使用相同（别名化的）的主机名；例如，一个公司的 Web 服务器和邮件服务器都能叫作 enterprise. com。

- **负载分配**（load distribution）。DNS 也用于在冗余的服务器（如冗余的 Web 服务器等）之间进行负载分配。繁忙的站点（如 cnn. com）被冗余分布在多台服务器上，每台服务器均运行在不同的端系统上，每个都有着不同的 IP 地址。由于这些冗余的 Web 服务器，一个 IP 地址集合因此与同一个规范主机名相联系。DNS 数据库中存储着这些 IP 地址集合。当客户对映射到某地址集合的名字发出一个 DNS 请求时，该服务器用 IP 地址的整个集合进行响应，但在每个回答中循环这些地址次序。因为客户通常总是向 IP 地址排在最前面的服务器发送 HTTP 请求报文，所以 DNS 就在所有这些冗余的 Web 服务器之间循环分配了负载。DNS 的循环同样可以用于邮件服务器，因此，多个邮件服务器可以具有相同的别名。一些内容分发公司如 Akamai 也以更加复杂的方式使用 DNS [Dilley 2002]，以提供 Web 内容分发（参见 2. 6. 3 节）。

DNS 由 RFC 1034 和 RFC 1035 定义，并且在几个附加的 RFC 中进行了更新。DNS 是一个复杂的系统，我们在这里只是就其运行的主要方面进行学习。感兴趣的读者可以参考这些 RFC 文档和 Albitz 和 Liu 写的书 [Albitz 1993]；亦可参阅文章 [Mockapetris 1998] 和 [Mockapetris 2005]，其中 [Mockapetris 1998] 是回顾性的文章，它提供了 DNS 组成和工作原理的精细的描述。

2. 4. 2 DNS 工作机理概述

下面给出一个 DNS 工作过程的总体概括，我们的讨论将集中在主机名到 IP 地址转换服务方面。

假设运行在用户主机上的某些应用程序（如 Web 浏览器或邮件阅读器）需要将主机名转换为 IP 地址。这些应用程序将调用 DNS 的客户端，并指明需要被转换的主机名（在很多基于 UNIX 的机器上，应用程序为了执行这种转换需要调用函数 gethostbyname()）。用户主机上的 DNS 接收到后，向网络中发送一个 DNS 查询报文。所有的 DNS 请求和回答报文使用 UDP 数据报经端口 53 发送。经过若干毫秒到若干秒的时延后，用户主机上的 DNS 接收到一个提供所希望映射的 DNS 回答报文。这个映射结果则被传递到调用 DNS 的应用程序。因此，从用户主机上调用应用程序的角度看，DNS 是一个提供简单、直接的转换服务的黑盒子。但事实上，实现这个服务的黑盒子非常复杂，它由分布于全球的大量 DNS 服务器以及定义了 DNS 服务器与查询主机通信方式的应用层协议组成。

DNS 的一种简单设计是在因特网上只使用一个 DNS 服务器，该服务器包含所有的映射。在这种集中式设计中，客户直接将所有查询直接发往单一的 DNS 服务器，同时该 DNS 服务器直接对所有的查询客户做出响应。尽管这种设计的简单性非常具有吸引力，但它不适用于当今的因特网，因为因特网有着数量巨大（并持续增长）的主机。这种集中式设计的问题包括：

- **单点故障**（a single point of failure）。如果该 DNS 服务器崩溃，整个因特网随之瘫痪！
- **通信容量**（traffic volume）。单个 DNS 服务器不得不处理所有的 DNS 查询（用于为上亿台主机产生的所有 HTTP 请求报文和电子邮件报文服务）。

- **远距离的集中式数据库**（distant centralized database）。单个 DNS 服务器不可能"邻近"所有查询客户。如果我们将单台 DNS 服务器放在纽约市，那么所有来自澳大利亚的查询必须传播到地球的另一边，中间也许还要经过低速和拥塞的链路。这将导致严重的时延。
- **维护**（maintenance）。单个 DNS 服务器将不得不为所有的因特网主机保留记录。这不仅将使这个中央数据库庞大，而且它还不得不为解决每个新添加的主机而频繁更新。

总的来说，在单一 DNS 服务器上运行集中式数据库完全没有可扩展能力。因此，DNS 采用了分布式的设计方案。事实上，DNS 是一个在因特网上实现分布式数据库的精彩范例。

1. 分布式、层次数据库

为了处理扩展性问题，DNS 使用了大量的 DNS 服务器，它们以层次方式组织，并且分布在全世界范围内。没有一台 DNS 服务器拥有因特网上所有主机的映射。相反，这些映射分布在所有的 DNS 服务器上。大致说来，有 3 种类型的 DNS 服务器：根 DNS 服务器、顶级域（Top-Level Domain，TLD）DNS 服务器和权威 DNS 服务器。这些服务器以图 2-17 中所示的层次结构组织起来。为了理解这 3 种类型的 DNS 服务器交互的方式，假定一个 DNS 客户要决定主机名 www. amazon. com 的 IP 地址。粗略说来，将发生下列事件。客户首先与根服务器之一联系，它将返回顶级域名 com 的 TLD 服务器的 IP 地址。该客户则与这些 TLD 服务器之一联系，它将为 amazon. com 返回权威服务器的 IP 地址。最后，该客户与 amazon. com 权威服务器之一联系，它为主机名 www. amazon. com 返回其 IP 地址。我们将很快更为详细地考察 DNS 查找过程。不过我们先仔细看一下这 3 种类型的 DNS 服务器。



图 2-17　部分 DNS 服务器的层次结构

- **根 DNS 服务器**。有 400 多个根名字服务器遍及全世界。这些根名字服务器由 13 个不同的组织管理。根名字服务器的全部清单连同管理它们的组织及其 IP 地址可以在［Root Servers 2016］中找到。根名字服务器提供 TLD 服务器的 IP 地址。
- **顶级域（DNS）服务器**。对于每个顶级域（如 com、org、net、edu 和 gov）和所有国家的顶级域（如 uk、fr、ca 和 jp），都有 TLD 服务器（或服务器集群）。Verisign Global Registry Services 公司维护 com 顶级域的 TLD 服务器，Educause 公司维护 edu 顶级域的 TLD 服务器。支持 TLD 的网络基础设施可能是大而复杂的，［Osterweil 2012］对 Verisign 网络进行了很好的概述。所有顶级域的列表参见［TLD list

2016]。TLD 服务器提供了权威 DNS 服务器的 IP 地址。

- **权威 DNS 服务器**。在因特网上具有公共可访问主机（如 Web 服务器和邮件服务器）的每个组织机构必须提供公共可访问的 DNS 记录，这些记录将这些主机的名字映射为 IP 地址。一个组织机构的权威 DNS 服务器收藏了这些 DNS 记录。一个组织机构能够选择实现它自己的权威 DNS 服务器以保存这些记录；另一种方法是，该组织能够支付费用，让这些记录存储在某个服务提供商的一个权威 DNS 服务器中。多数大学和大公司实现和维护它们自己基本和辅助（备份）的权威 DNS 服务器。

根、TLD 和权威 DNS 服务器都处在该 DNS 服务器的层次结构中，如图 2-17 所示。还有另一类重要的 DNS 服务器，称为**本地 DNS 服务器**（local DNS server）。严格说来，一个本地 DNS 服务器并不属于该服务器的层次结构，但它对 DNS 层次结构是至关重要的。每个 ISP（如一个居民区的 ISP 或一个机构的 ISP）都有一台本地 DNS 服务器（也叫默认名字服务器）。当主机与某个 ISP 连接时，该 ISP 提供一台主机的 IP 地址，该主机具有一台或多台其本地 DNS 服务器的 IP 地址（通常通过 DHCP，将在第 4 章中讨论）。通过访问 Windows 或 UNIX 的网络状态窗口，用户能够容易地确定他的本地 DNS 服务器的 IP 地址。主机的本地 DNS 服务器通常“邻近”本主机。对某机构 ISP 而言，本地 DNS 服务器可能就与主机在同一个局域网中；对于某居民区 ISP 来说，本地 DNS 服务器通常与主机相隔不超过几台路由器。当主机发出 DNS 请求时，该请求被发往本地 DNS 服务器，它起着代理的作用，并将该请求转发到 DNS 服务器层次结构中，我们下面将更为详细地讨论。

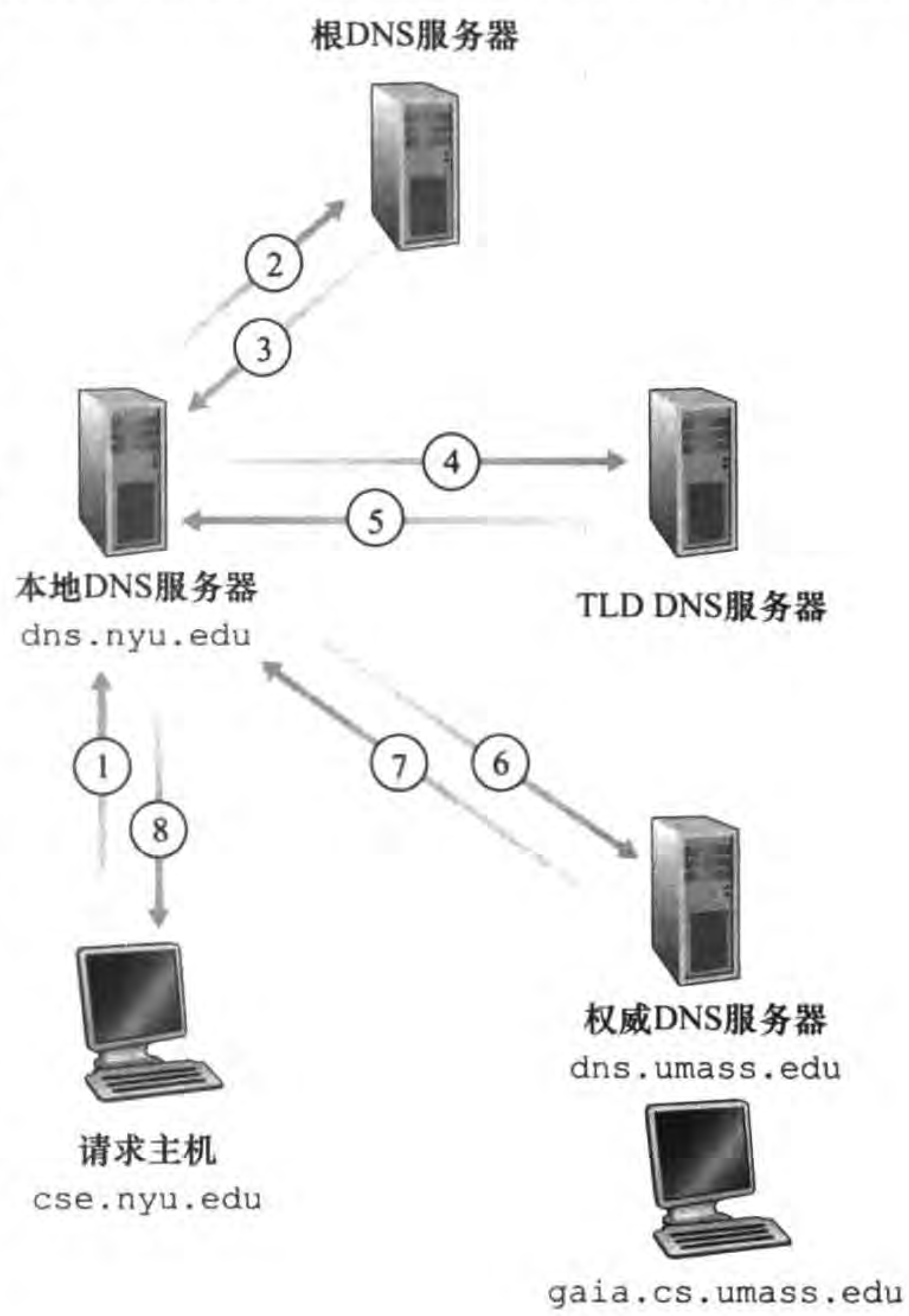


图 2-18 各种 DNS 服务器的交互

我们来看一个简单的例子，假设主机 cse. nyu. edu 想知道主机 gaia. cs. umass. edu 的 IP 地址。同时假设纽约大学（NYU）的 cse. nyu. edu 主机的本地 DNS 服务器为 dns. nyu. edu，并且 gaia. cs. umass. edu 的权威 DNS 服务器为 dns. umass. edu。如图 2-18 所示，主机 cse. nyu. edu 首先向它的本地 DNS 服务器 dns. nyu. edu 发送一个 DNS 查询报文。该查询报文含有被转换的主机名 gaia. cs. umass. edu。本地 DNS 服务器将该报文转发到根 DNS 服务器。该根 DNS 服务器注意到其 edu 前缀并向本地 DNS 服务器返回负责 edu 的 TLD 的 IP 地址列表。该本地 DNS 服务器则再次向这些 TLD 服务器之一发送查询报文。该 TLD 服务器注意到 umass. edu 前缀，并用权威 DNS 服务器的 IP 地址进行响应，该权威 DNS 服务器是负责马萨诸塞大学的 dns. umass. edu。最后，本地 DNS 服务器直接向 dns. umass. edu 重发查询报

文，dns. umass. edu 用 gaia. cs. umass. edu 的 IP 地址进行响应。注意到在本例中，为了获得一台主机名的映射，共发送了 8 份 DNS 报文：4 份查询报文和 4 份回答报文！我们将很快明白利用 DNS 缓存减少这种查询流量的方法。

我们前面的例子假设了 TLD 服务器知道用于主机的权威 DNS 服务器的 IP 地址。一般而言，这种假设并不总是正确的。相反，TLD 服务器只是知道中间的某个 DNS 服务器，该中间 DNS 服务器依次才能知道用于该主机的权威 DNS 服务器。例如，再次假设马萨诸塞大学有一台用于本大学的 DNS 服务器，它称为 dns. umass. edu。同时假设该大学的每个系都有自己的 DNS 服务器，每个系的 DNS 服务器是本系所有主机的权威服务器。在这种情况下，当中间 DNS 服务器 dns. umass. edu 收到了对某主机的请求时，该主机名是以 cs. umass. edu 结尾，它向 dns. nyu. edu 返回 dns. cs. umass. edu 的 IP 地址，后者是所有以 cs. umass. edu 结尾的主机的权威服务器。本地 DNS 服务器 dns. nyu. edu 则向权威 DNS 服务器发送查询，该权威 DNS 服务器向本地 DNS 服务器返回所希望的映射，该本地服务器依次向请求主机返回该映射。在这个例子中，共发送了 10 份 DNS 报文！

图 2-18 所示的例子利用了**递归查询**（recursive query）和**迭代查询**（iterative query）。从 cse. nyu. edu 到 dns. nyu. edu 发出的查询是递归查询，因为该查询以自己的名义请求 dns. nyu. edu 来获得该映射。而后继的 3 个查询是迭代查询，因为所有的回答都是直接返回给 dns. nyu. edu。从理论上讲，任何 DNS 查询既可以是迭代的也能是递归的。例如，图 2-19 显示了一条 DNS 查询链，其中的所有查询都是递归的。实践中，查询通常遵循图 2-18 中的模式：从请求主机到本地 DNS 服务器的查询是递归的，其余的查询是迭代的。

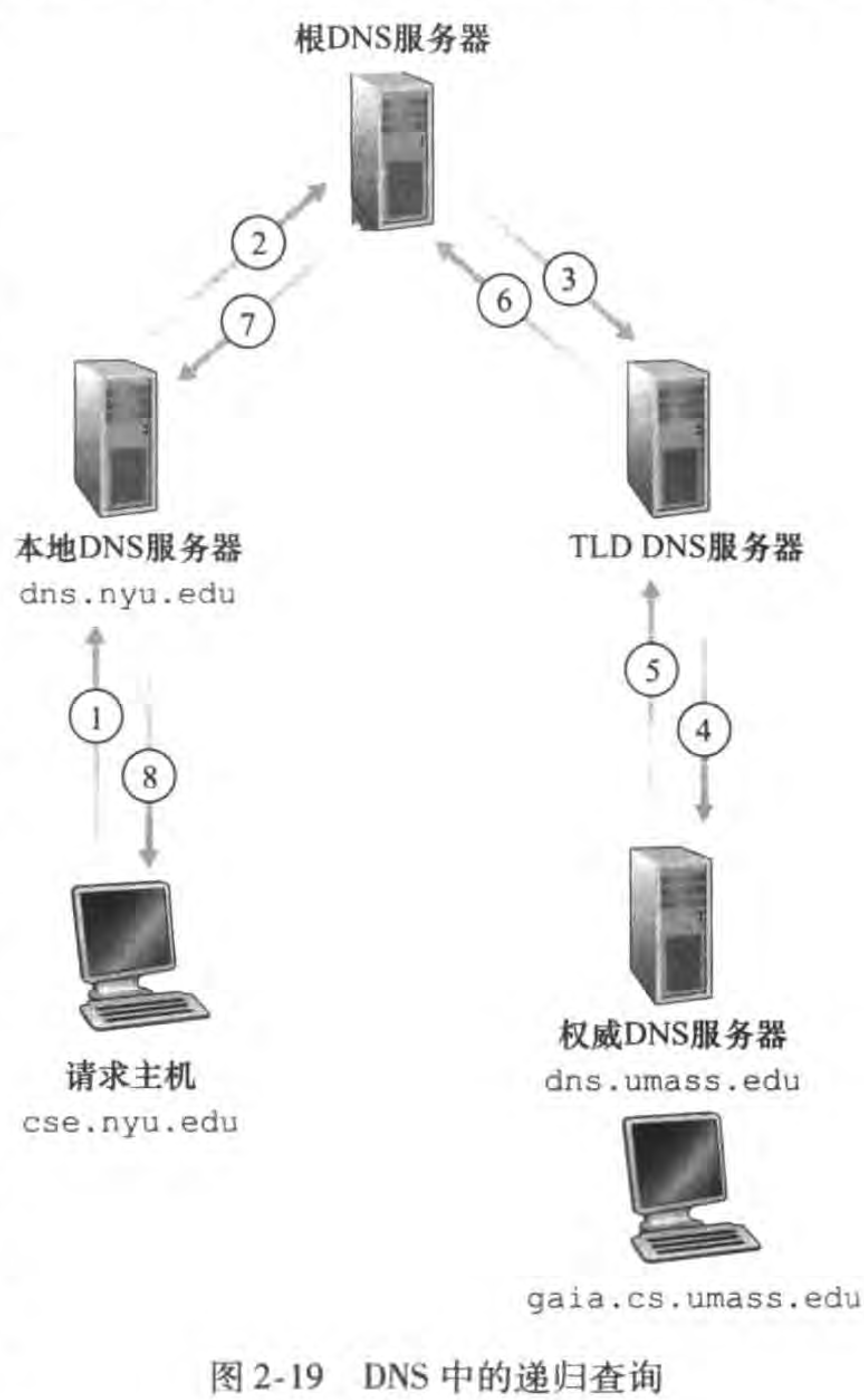


图 2-19　DNS 中的递归查询

2. DNS 缓存

至此我们的讨论一直忽略了 DNS 系统的一个非常重要特色：**DNS 缓存**（DNS caching）。实际上，为了改善时延性能并减少在因特网上到处传输的 DNS 报文数量，DNS 广泛使用了缓存技术。DNS 缓存的原理非常简单。在一个请求链中，当某 DNS 服务器接收一个 DNS 回答（例如，包含某主机名到 IP 地址的映射）时，它能将映射缓存在本地存储器中。例如，在图 2-18 中，每当本地 DNS 服务器 dns. nyu. edu 从某个 DNS 服务器接收到一个回答，它能够缓存包含在该回答中的任何信息。如果在 DNS 服务器中缓存了一台主机名/IP 地址对，另一个对相同主机名的查询到达该 DNS 服务器时，该 DNS 服务器就能够提供所要求的 IP 地址，即使它不是该主机名的权威服务器。由于主机和主机名与 IP 地址间的

映射并不是永久的，DNS 服务器在一段时间后（通常设置为两天）将丢弃缓存的信息。

举一个例子，假定主机 apricot. nyu. edu 向 dns. nyu. edu 查询主机名 cnn. com 的 IP 地址。此后，假定过了几个小时，纽约大学的另外一台主机如 kiwi. nyu. edu 也向 dns. nyu. edu 查询相同的主机名。因为有了缓存，该本地 DNS 服务器可以立即返回 cnn. com 的 IP 地址，而不必查询任何其他 DNS 服务器。本地 DNS 服务器也能够缓存 TLD 服务器的 IP 地址，因而允许本地 DNS 绕过查询链中的根 DNS 服务器。事实上，因为缓存，除了少数 DNS 查询以外，根服务器被绕过了。

2.4.3 DNS 记录和报文

共同实现 DNS 分布式数据库的所有 DNS 服务器存储了**资源记录**（Resource Record, RR），RR 提供了主机名到 IP 地址的映射。每个 DNS 回答报文包含了一条或多条资源记录。在本小节以及后续小节中，我们概要地介绍 DNS 资源记录和报文，更详细的信息可以在［Albitz 1993］或有关 DNS 的 RFC 文档［RFC 1034；RFC 1035］中找到。

资源记录是一个包含了下列字段的 4 元组：

```
(Name, Value, Type, TTL)
```

TTL 是该记录的生存时间，它决定了资源记录应当从缓存中删除的时间。在下面给出的记录例子中，我们忽略掉 TTL 字段。Name 和 Value 的值取决于 Type：

- 如果 Type = A，则 Name 是主机名，Value 是该主机名对应的 IP 地址。因此，一条类型为 A 的资源记录提供了标准的主机名到 IP 地址的映射。例如（relay1. bar. foo. com，145. 37. 93. 126，A）就是一条类型 A 记录。
- 如果 Type = NS，则 Name 是个域（如 foo. com），而 Value 是个知道如何获得该域中主机 IP 地址的权威 DNS 服务器的主机名。这个记录用于沿着查询链来路由 DNS 查询。例如（foo. com，dns. foo. com，NS）就是一条类型为 NS 的记录。
- 如果 Type = CNAME，则 Value 是别名为 Name 的主机对应的规范主机名。该记录能够向查询的主机提供一个主机名对应的规范主机名，例如（foo. com，relay1. bar. foo. com，CNAME）就是一条 CNAME 类型的记录。
- 如果 Type = MX，则 Value 是个别名为 Name 的邮件服务器的规范主机名。举例来说，（foo. com，mail. bar. foo. com，MX）就是一条 MX 记录。MX 记录允许邮件服务器主机名具有简单的别名。值得注意的是，通过使用 MX 记录，一个公司的邮件服务器和其他服务器（如它的 Web 服务器）可以使用相同的别名。为了获得邮件服务器的规范主机名，DNS 客户应当请求一条 MX 记录；而为了获得其他服务器的规范主机名，DNS 客户应当请求 CNAME 记录。

如果一台 DNS 服务器是用于某特定主机名的权威 DNS 服务器，那么该 DNS 服务器会有一条包含用于该主机名的类型 A 记录（即使该 DNS 服务器不是其权威 DNS 服务器，它也可能在缓存中包含有一条类型 A 记录）。如果服务器不是用于某主机名的权威服务器，那么该服务器将包含一条类型 NS 记录，该记录对应于包含主机名的域；它还将包括一条类型 A 记录，该记录提供了在 NS 记录的 Value 字段中的 DNS 服务器的 IP 地址。举例来说，假设一台 edu TLD 服务器不是主机 gaia. cs. umass. edu 的权威 DNS 服务器，则该服务器将包含一条包括主机 cs. umass. edu 的域记录，如（umass. edu，dns. umass. edu，NS）；该 edu TLD 服务器还将包含一条类型 A 记录，如（dns. umass. edu，128. 119. 40. 111，A），

该记录将名字 dns. umass. edu 映射为一个 IP 地址。

1. DNS 报文

在本节前面，我们提到了 DNS 查询和回答报文。DNS 只有这两种报文，并且，查询和回答报文有着相同的格式，如图 2-20 所示。DNS 报文中各字段的语义如下：

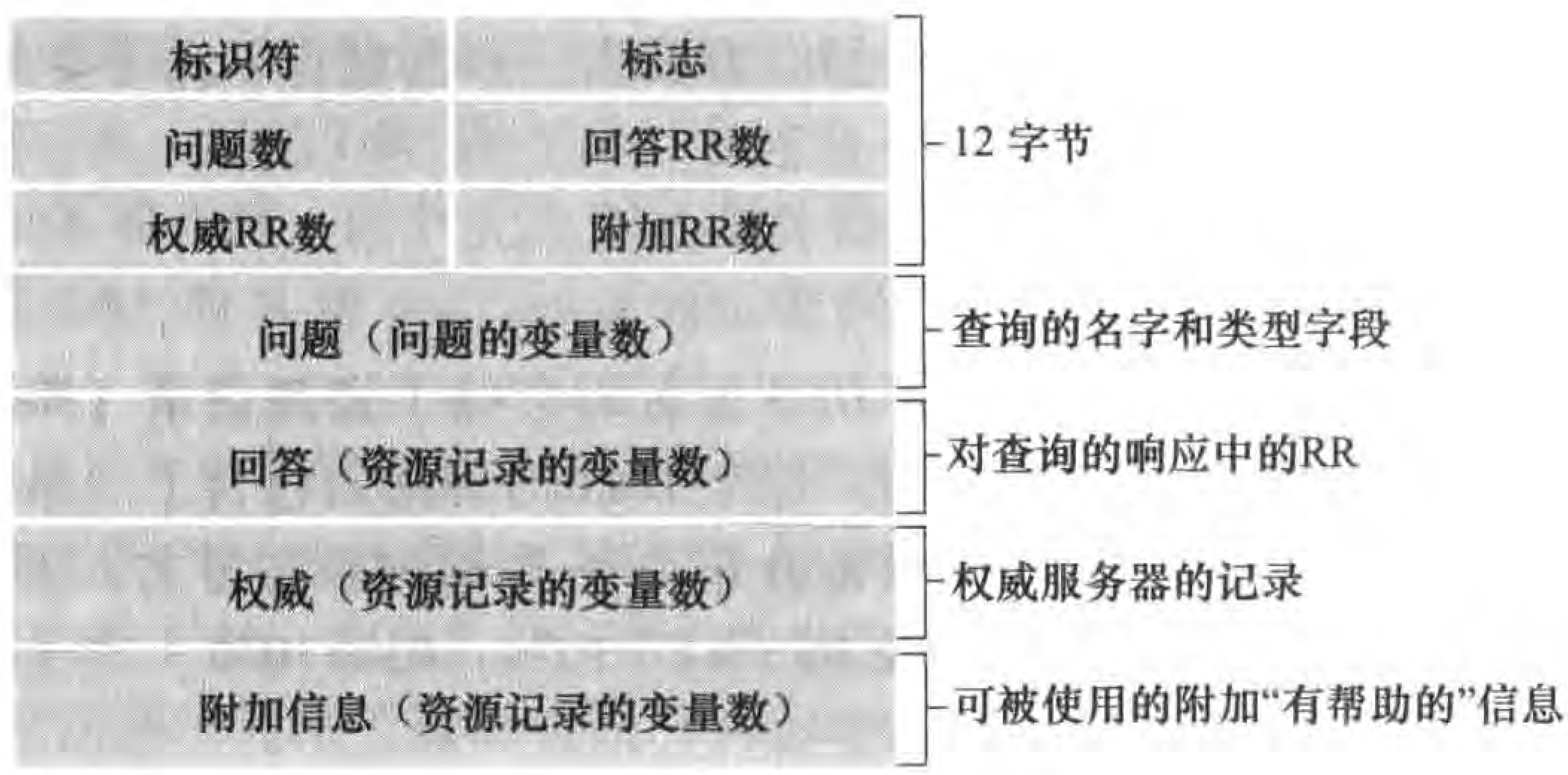


图 2-20 DNS 报文格式

- 前 12 个字节是首部区域，其中有几个字段。第一个字段（标识符）是一个 16 比特的数，用于标识该查询。这个标识符会被复制到对查询的回答报文中，以便让客户用它来匹配发送的请求和接收到的回答。标志字段中含有若干标志。1 比特的“查询/回答”标志位指出报文是查询报文（0）还是回答报文（1）。当某 DNS 服务器是所请求名字的权威 DNS 服务器时，1 比特的“权威的”标志位被置在回答报文中。如果客户（主机或者 DNS 服务器）在该 DNS 服务器没有某记录时希望它执行递归查询，将设置 1 比特的“希望递归”标志位。如果该 DNS 服务器支持递归查询，在它的回答报文中会对 1 比特的“递归可用”标志位置位。在该首部中，还有 4 个有关数量的字段，这些字段指出了在首部后的 4 类数据区域出现的数量。
- 问题区域包含着正在进行的查询信息。该区域包括：①名字字段，包含正在被查询的主机名字；②类型字段，指出有关该名字的正被询问的问题类型，例如主机地址是与一个名字相关联（类型 A）还是与某个名字的邮件服务器相关联（类型 MX）。
- 在来自 DNS 服务器的回答中，回答区域包含了对最初请求的名字的资源记录。前面讲过每个资源记录中有 Type（如 A、NS、CNAME 和 MX）字段、Value 字段和 TTL 字段。在回答报文的回答区域中可以包含多条 RR，因此一个主机名能够有多个 IP 地址（例如，就像本节前面讨论的冗余 Web 服务器）。
- 权威区域包含了其他权威服务器的记录。
- 附加区域包含了其他有帮助的记录。例如，对于一个 MX 请求的回答报文的回答区域包含了一条资源记录，该记录提供了邮件服务器的规范主机名。该附加区域包含一个类型 A 记录，该记录提供了用于该邮件服务器的规范主机名的 IP 地址。

你愿意从正在工作的主机直接向某些 DNS 服务器发送一个 DNS 查询报文吗？使用 **nslookup 程序**（nslookup program）能够容易地做到这一点，这对于多数 Windows 和 UNIX 平台，nslookup 程序是可用的。例如，从一台 Windows 主机打开命令提示符界面，直接键入“nslookup”即可调用该 nslookup 程序。在调用 nslookup 后，你能够向任何 DNS 服务器

(根、TLD 或权威) 发送 DNS 查询。在接收到来自 DNS 服务器的回答后，nslookup 将显示包括在该回答中的记录（以人可读的格式）。从你自己的主机运行 nslookup 还有一种方法，即访问允许你远程应用 nslookup 的许多 Web 站点之一（在一个搜索引擎中键入 "nslookup" 就能够得到这些站点中的一个）。本章最后的 DNS Wireshark 实验将使你更为详细地研究 DNS。

2. 在 DNS 数据库中插入记录

上面的讨论只是关注如何从 DNS 数据库中取数据。你可能想知道这些数据最初是怎么进入数据库中的。我们在一个特定的例子中看看这是如何完成的。假定你刚刚创建一个称为网络乌托邦（Network Utopia）的令人兴奋的新创业公司。你必定要做的第一件事是在注册登记机构注册域名 networkutopia. com。**注册登记机构**（registrar）是一个商业实体，它验证该域名的唯一性，将该域名输入 DNS 数据库（如下面所讨论的那样），对提供的服务收取少量费用。1999 年前，唯一的注册登记机构是 Nework Solution，它独家经营对于 com、net 和 org 域名的注册。但是现在有许多注册登记机构竞争客户，因特网名字和地址分配机构（Internet Corporation for Assigned Names and Numbers，ICANN）向各种注册登记机构授权。在 http://www. internic. net 上可以找到授权的注册登记机构的列表。

当你向某些注册登记机构注册域名 networkutopia. com 时，需要向该机构提供你的基本和辅助权威 DNS 服务器的名字和 IP 地址。假定该名字和 IP 地址是 dns1. networkutopia. com 和 dns2. networkutopia. com 及 212. 212. 212. 1 和 212. 212. 212. 2。对这两个权威 DNS 服务器的每一个，该注册登记机构确保将一个类型 NS 和一个类型 A 的记录输入 TLD com 服务器。特别是对于用于 networkutopia. com 的基本权威服务器，该注册登记机构将下列两条资源记录插入该 DNS 系统中：

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

你还必须确保用于 Web 服务器 www. networkutopia. com 的类型 A 资源记录和用于邮件服务器 mail. networkutopia. com 的类型 MX 资源记录被输入你的权威 DNS 服务器中。（直到最近，每台 DNS 服务器中的内容都是静态配置的，例如来自系统管理员创建的配置文件。最近，在 DNS 协议中添加了一个更新（UPDATE）选项，允许通过 DNS 报文对数据库中的内容进行动态添加或者删除。[RFC 2136] 和 [RFC 3007] 定义了 DNS 动态更新。）

一旦完成所有这些步骤，人们将能够访问你的 Web 站点，并向你公司的雇员发送电子邮件。我们通过验证该说法的正确性来总结 DNS 的讨论。这种验证也有助于充实我们已经学到的 DNS 知识。假定在澳大利亚的 Alice 要观看 www. networkutopia. com 的 Web 页面。如前面所讨论，她的主机将首先向其本地 DNS 服务器发送请求。该本地服务器接着则联系一个 TLD com 服务器。（如果 TLD com 服务器的地址没有被缓存，该本地 DNS 服务器也将必须与根 DNS 服务器相联系。）该 TLD 服务器包含前面列出的类型 NS 和类型 A 资源记录，因为注册登记机构将这些资源记录插入所有的 TLD com 服务器。该 TLD com 服务器向 Alice 的本地 DNS 服务器发送一个回答，该回答包含了这两条资源记录。该本地 DNS 服务器则向 212. 212. 212. 1 发送一个 DNS 查询，请求对应于 www. networkutopia. com 的类型 A 记录。该记录提供了所希望的 Web 服务器的 IP 地址，如 212. 212. 71. 4，本地 DNS 服务器将该地址回传给 Alice 的主机。Alice 的浏览器此时能够向主机 212. 212. 71. 4 发起一个 TCP 连接，并在该连接上发送一个 HTTP 请求。当一个人在网上冲浪时，有比满足眼球更

多的事情在进行！

关注安全性

DNS 脆弱性

我们已经看到 DNS 是因特网基础设施的一个至关重要的组件，对于包括 Web、电子邮件等的许多重要的服务，没有它都不能正常工作。因此，我们自然要问，DNS 怎么能够被攻击呢？DNS 是一个易受攻击的目标吗？它是将会被淘汰的服务吗？大多数因特网应用会随同它一起无法工作吗？

想到的第一种针对 DNS 服务的攻击是分布式拒绝服务（DDoS）带宽洪泛攻击（参见 1.6 节）。例如，某攻击者能够试图向每个 DNS 根服务器发送大量的分组，使得大多数合法 DNS 请求得不到回答。这种对 DNS 根服务器的 DDoS 大规模攻击实际发生在 2002 年 10 月 21 日。在这次攻击中，该攻击者利用一个僵尸网络向 13 个 DNS 根服务器中的每个都发送了大批的 ICMP ping 报文负载。（5.6 节中讨论了 ICMP 报文。此时，知道 ICMP 分组是特殊类型的 IP 数据报就可以了。）幸运的是，这种大规模攻击所带来的损害很小，对用户的因特网体验几乎没有或根本没有影响。攻击者的确成功地将大量的分组指向了根服务器，但许多 DNS 根服务器受到了分组过滤器的保护，配置的分组过滤器阻挡了所有指向根服务器的 ICMP ping 报文。这些被保护的服务器因此未受伤害并且与平常一样发挥着作用。此外，大多数本地 DNS 服务器缓存了顶级域名服务器的 IP 地址，使得这些请求过程通常绕过了 DNS 根服务器。

对 DNS 的潜在更为有效的 DDoS 攻击将是向顶级域名服务器（例如向所有处理.com 域的顶级域名服务器）发送大量的 DNS 请求。过滤指向 DNS 服务器的 DNS 请求将更为困难，并且顶级域名服务器不像根服务器那样容易绕过。但是这种攻击的严重性通过本地 DNS 服务器中的缓存技术可将部分地被缓解。

DNS 能够潜在地以其他方式被攻击。在中间人攻击中，攻击者截获来自主机的请求并返回伪造的回答。在 DNS 毒害攻击中，攻击者向一台 DNS 服务器发送伪造的回答，诱使服务器在它的缓存中接收伪造的记录。这些攻击中的任一种，都能够将毫无疑虑的 Web 用户重定向到攻击者的 Web 站点。然而，这些攻击难以实现，因为它们要求截获分组或扼制住服务器［Skoudis 2006］。

总而言之，DNS 自身已经显示了对抗攻击的令人惊讶的健壮性。至今为止，还没有一个攻击已经成功地妨碍了 DNS 服务。

2.5　P2P 文件分发

在目前为止本章中描述的应用（包括 Web、电子邮件和 DNS）都采用了客户－服务器体系结构，极大地依赖于总是打开的基础设施服务器。2.1.1 节讲过，使用 P2P 体系结构，对总是打开的基础设施服务器有最小的（或者没有）依赖。与之相反，成对间歇连接的主机（称为对等方）彼此直接通信。这些对等方并不为服务提供商所拥有，而是受用户控制的桌面计算机和膝上计算机。

在本节中我们将研究一个非常自然的 P2P 应用，即从单一服务器向大量主机（称为对

等方）分发一个大文件。该文件也许是一个新版的 Linux 操作系统，对于现有操作系统或应用程序的一个软件补丁，一个 MP3 音乐文件，或一个 MPEG 视频文件。在客户 - 服务器文件分发中，该服务器必须向每个对等方发送该文件的一个副本，即服务器承受了极大的负担，并且消耗了大量的服务器带宽。在 P2P 文件分发中，每个对等方能够向任何其他对等方重新分发它已经收到的该文件的任何部分，从而在分发过程中协助该服务器。到 2016 年止，最为流行的 P2P 文件分发协议是 BitTorrent。该应用程序最初由 Bram Cohen 所研发，现在有许多不同的独立且符合 BitTorrent 协议的 BitTorrent 客户，就像有许多符合 HTTP 协议的 Web 浏览器客户一样。在下面的小节中，我们首先考察在文件分发环境中的 P2P 体系结构的自扩展性。然后我们更为详细地描述 BitTorrent，突出它的最为重要的特性和特色。

1. P2P 体系结构的扩展性

为了将客户 - 服务器体系结构与 P2P 体系结构进行比较，阐述 P2P 的内在自扩展性，我们现在考虑一个用于两种体系结构类型的简单定量模型，将一个文件分发给一个固定对等方集合。如图 2-21 所示，服务器和对等方使用接入链路与因特网相连。其中 u_s 表示服务器接入链路的上载速率，u_i 表示第 i 对等方接入链路的上载速率，d_i 表示了第 i 对等方接入链路的下载速率。还用 F 表示被分发的文件长度（以比特计），N 表示要获得的该文件副本的对等方的数量。**分发时间**（distribution time）是所有 N 个对等方得到该文件的副本所需要的时间。在下面分析分发时间的过程中，我们对客户 - 服务器和 P2P 体系结构做了简化（并且通常是准确的［Akella 2003］）的假设，即因特网核心具有足够的带宽，这意味着所有瓶颈都在网络接入链路。我们还假设服务器和客户没有参与任何其他网络应用，因此它们的所有上传和下载访问带宽能被全部用于分发该文件。

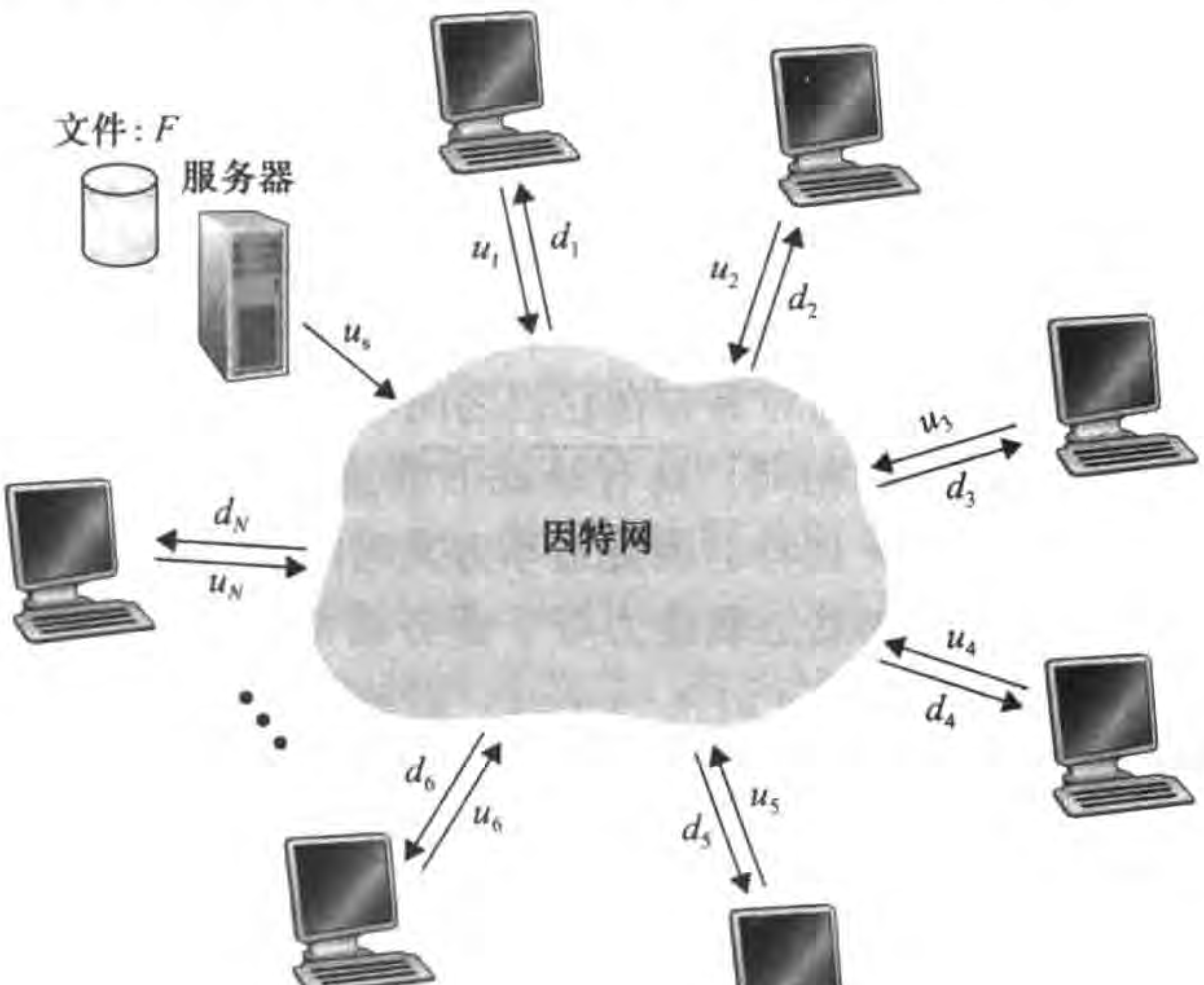


图 2-21 文件分发问题的示例图

我们首先来确定对于客户 - 服务器体系结构的分发时间，我们将其表示为 D_{cs}。在客户 - 服务器体系结构中，没有对等方参与来帮助分发文件。我们做下列观察：

- 服务器必须向 N 个对等方的每个传输该文件的一个副本。因此该服务器必须传输 NF 比特。因为该服务器的上载速率是 u_s，分发该文件的时间必定是至少为 NF/u_s。
- 令 d_{min} 表示具有最小下载速率的对等方的下载速率，即 $d_{min} = \min\{d_1, d_p, \cdots, d_N\}$。具有最小下载速率的对等方不可能在少于 F/d_{min} 秒时间内获得该文件的所有 F 比特。因此最小分发时间至少为 F/d_{min}。

将这两个观察放在一起，我们得到

$$D_{cs} \geqslant \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

该式提供了对于客户-服务器体系结构的最小分发时间的下界。在课后习题中将请你给出服务器能够调度它的传输以便实际取得该下界的方法。因此我们取上面提供的这个下界作为实际发送时间，即

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\} \tag{2-1}$$

我们从式（2-1）看到，对足够大的 N，客户-服务器分发时间由 NF/u_s 确定。所以，该分发时间随着对等方 N 的数量线性地增加。因此举例来说，如果从某星期到下星期对等方的数量从1000增加了1000倍，到了100万，将该文件分发到所有对等方所需要的时间就要增加1000倍。

我们现在来对P2P体系结构进行简单的分析，其中每个对等方能够帮助服务器分发该文件。特别是，当一个对等方接收到某些文件数据，它能够使用自己的上载能力重新将数据分发给其他对等方。计算P2P体系结构的分发时间在某种程度上比计算客户-服务器体系结构的更为复杂，因为分发时间取决于每个对等方如何向其他对等方分发该文件的各个部分。无论如何，能够得到对该最小分发时间的一个简单表示式［Kumar 2006］。至此，我们先做下列观察：

- 在分发的开始，只有服务器具有文件。为了使社区的这些对等方得到该文件，该服务器必须经其接入链路至少发送该文件的每个比特一次。因此，最小分发时间至少是 F/u_s。（与客户-服务器方案不同，由服务器发送过一次的比特可能不必由该服务器再次发送，因为对等方在它们之间可以重新分发这些比特。）
- 与客户-服务器体系结构相同，具有最低下载速率的对等方不能够以小于 F/d_{min} 秒的分发时间获得所有 F 比特。因此最小分发时间至少为 F/d_{min}。
- 最后，观察到系统整体的总上载能力等于服务器的上载速率加上每个单独的对等方的上载速率，即 $u_{total} = u_s + u_1 + \cdots + u_N$。系统必须向这 N 个对等方的每个交付（上载）F 比特，因此总共交付 NF 比特。这不能以快于 u_{total} 的速率完成。因此，最小的分发时间也至少是 $NF/(u_s + u_1 + \cdots + u_N)$。

将这三个观察放在一起，我们获得了对P2P的最小分发时间，表示为 D_{P2P}。

$$D_{P2P} \geqslant \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\} \tag{2-2}$$

式（2-2）提供了对于P2P体系结构的最小分发时间的下界。这说明，如果我们认为一旦每个对等方接收到一个比特就能够重分发一个比特的话，则存在一个重新分发方案能实际取得这种下界［Kumar 2006］。（我们将在课后习题中证明该结果的一种特情形。）实

际上，被分发的是文件块而不是一个个比特。式（2-2）能够作为实际最小分发时间的很好近似。因此，我们取由式（2-2）提供的下界作为实际的最小分发时间，即

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\} \tag{2-3}$$

图 2-22 比较了客户 - 服务器和 P2P 体系结构的最小分发时间，其中假定所有的对等方具有相同的上载速率 u。在图 2-22 中，我们已经设置了 $F/u = 1$ 小时，$u_s = 10u$，$d_{min} \geq u_s$。因此，在一个小时中一个对等方能够传输整个文件，该服务器的传输速率是对等方上载速率的 10 倍，并且（为了简化起见）对等方的下载速率被设置得足够大，使之不会产生影响。我们从图 2-22 中看到，对于客户 - 服务器体系结构，随着对等方数量的增加，分发时间呈线性增长并且没有界。然而，对于 P2P 体系结构，最小分发时间不仅总是小于客户 - 服务器体系结构的分发时间，并且对于任意的对等方数量 N，总是小于 1 小时。因此，具有 P2P 体系结构的应用程序能够是自扩展的。这种扩展性的直接成因是：对等方除了是比特的消费者外还是它们的重新分发者。

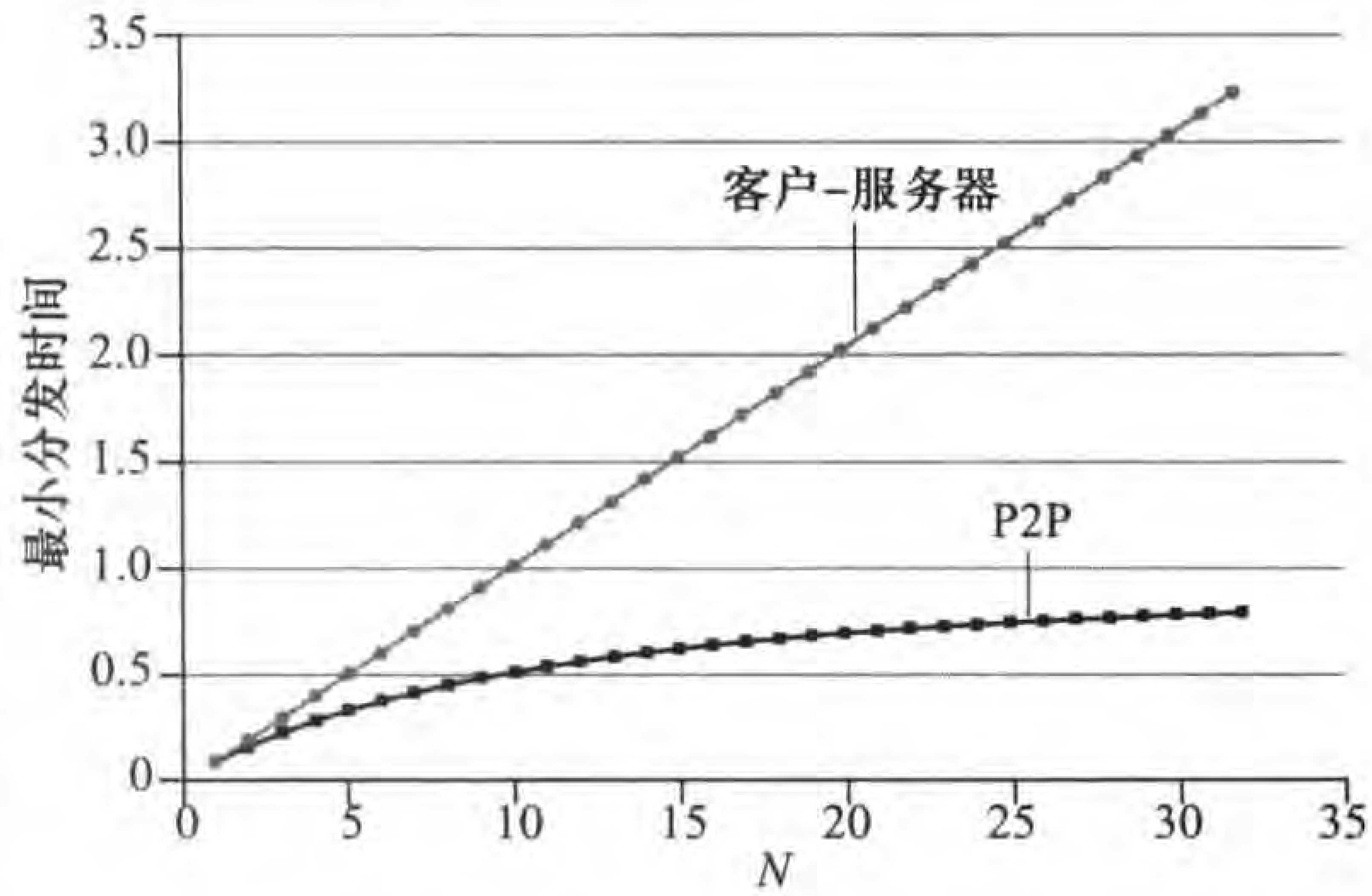


图 2-22 P2P 和客户 - 服务器体系结构的分发时间

2. BitTorrent

BitTorrent 是一种用于文件分发的流行 P2P 协议［Chao 2011］。用 BitTorrent 的术语来讲，参与一个特定文件分发的所有对等方的集合被称为一个*洪流*（torrent）。在一个洪流中的对等方彼此下载等长度的文件块（chunk），典型的块长度为 256KB。当一个对等方首次加入一个洪流时，它没有块。随着时间的流逝，它累积了越来越多的块。当它下载块时，也为其他对等方上载了多个块。一旦某对等方获得了整个文件，它也许（自私地）离开洪流，或（大公无私地）留在该洪流中并继续向其他对等方上载块。同时，任何对等方可能在任何时候仅具有块的子集就离开该洪流，并在以后重新加入该洪流中。

我们现在更为仔细地观察 BitTorrent 运行的过程。因为 BitTorrent 是一个相当复杂的协议，所以我们将仅描述它最重要的机制，而对某些细节视而不见；这将使得我们能够通过树木看森林。每个洪流具有一个基础设施节点，称为*追踪器*（tracker）。当一个对等方加入某洪流时，它向追踪器注册自己，并周期性地通知追踪器它仍在该洪流中。以这种方式，追踪器跟踪参与在洪流中的对等方。一个给定的洪流可能在任何时刻具有数以百计或数以千计的对等方。

如图 2-23 所示，当一个新的对等方 Alice 加入该洪流时，追踪器随机地从参与对等方的集合中选择对等方的一个子集（为了具体起见，设有 50 个对等方），并将这 50 个对等方的 IP 地址发送给 Alice。Alice 持有对等方的这张列表，试图与该列表上的所有对等方创建并行的 TCP 连接。我们称所有这样与 Alice 成功地创建一个 TCP 连接的对等方为“邻近对等方”（在图 2-23 中，Alice 显示了仅有三个邻近对等方。通常，她应当有更多的对等方）。随着时间的流逝，这些对等方中的某些可能离开，其他对等方（最初 50 个以外的）

可能试图与 Alice 创建 TCP 连接。因此一个对等方的邻近对等方将随时间而波动。

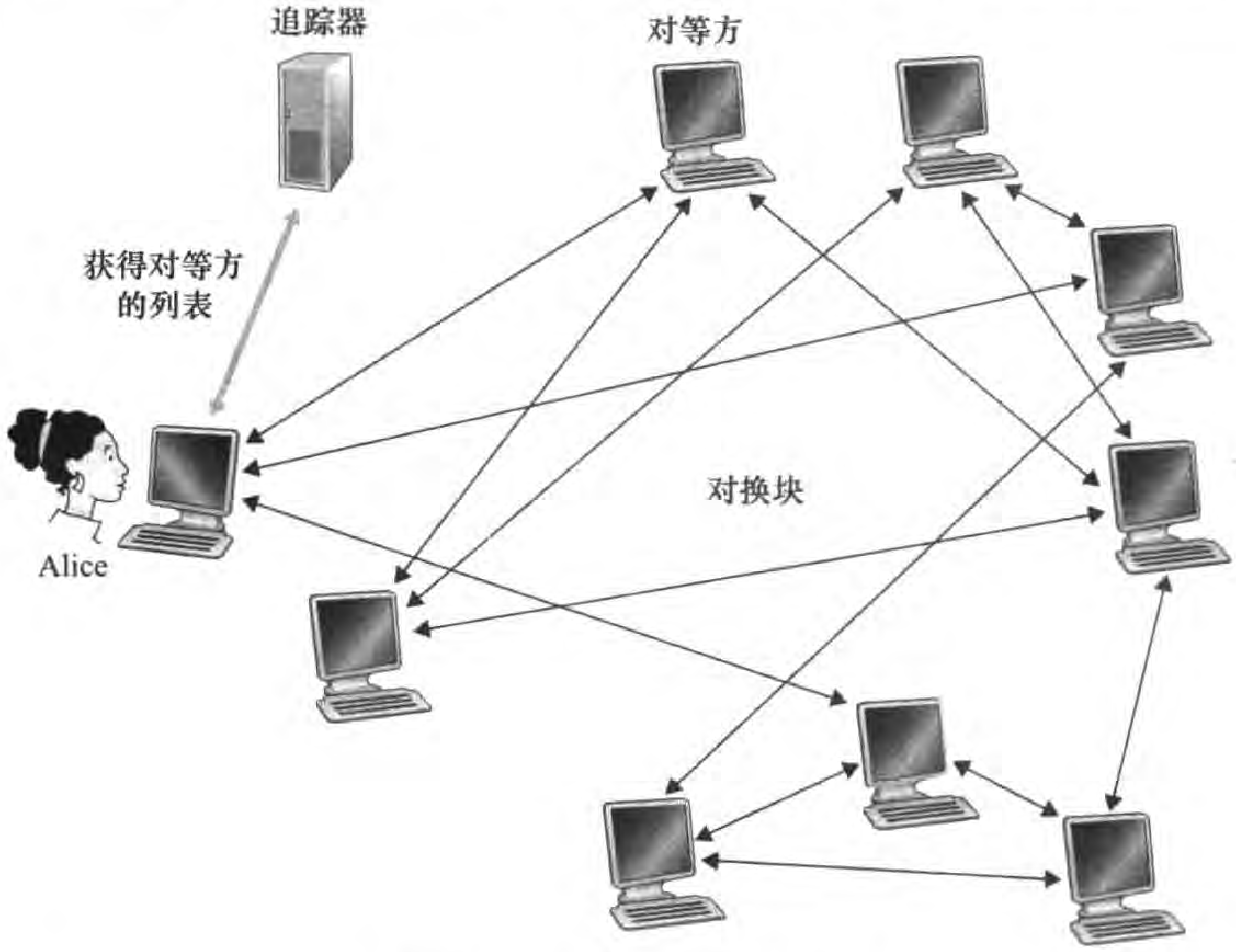


图 2-23　用 BitTorrent 分发文件

在任何给定的时间，每个对等方将具有来自该文件的块的子集，并且不同的对等方具有不同的子集。Alice 周期性地（经 TCP 连接）询问每个邻近对等方它们所具有的块列表。如果 Alice 具有 L 个不同的邻居，她将获得 L 个块列表。有了这个信息，Alice 将对她当前还没有的块发出请求（仍通过 TCP 连接）。

因此在任何给定的时刻，Alice 将具有块的子集并知道它的邻居具有哪些块。利用这些信息，Alice 将做出两个重要决定。第一，她应当从她的邻居请求哪些块呢？第二，她应当向哪些向她请求块的邻居发送块？在决定请求哪些块的过程中，Alice 使用一种称为**最稀缺优先**（rarest first）的技术。这种技术的思路是，针对她没有的块在她的邻居中决定最稀缺的块（最稀缺的块就是那些在她的邻居中副本数量最少的块），并首先请求那些最稀缺的块。这样，最稀缺块得到更为迅速的重新分发，其目标是（大致地）均衡每个块在洪流中的副本数量。

为了决定她响应哪个请求，BitTorrent 使用了一种机灵的对换算法。其基本想法是，Alice 根据当前能够以最高速率向她提供数据的邻居，给出其优先权。特别是，Alice 对于她的每个邻居都持续地测量接收到比特的速率，并确定以最高速率流入的 4 个邻居。每过 10 秒，她重新计算该速率并可能修改这 4 个对等方的集合。用 BitTorrent 术语来说，这 4 个对等方被称为**疏通**（unchoked）。重要的是，每过 30 秒，她也要随机地选择另外一个邻居并向其发送块。我们将这个被随机选择的对等方称为 Bob。因为 Alice 正在向 Bob 发送数据，她可能成为 Bob 前 4 位上载者之一，这样的话 Bob 将开始向 Alice 发送数据。如果 Bob 向 Alice 发送数据的速率足够高，Bob 接下来也能成为 Alice 的前 4 位上载者。换言之，每过 30 秒 Alice 将随机地选择一名新的对换伴侣并开始与那位伴侣进行对换。如果这两名对等方都满足此对换，它们将对方放入其前 4 位列表中并继续与对方进行对换，直到该对等方之一发现了一个更好的伴侣为止。这种效果是对等方能够以趋向于找到彼此的协调的

速率上载。随机选择邻居也允许新的对等方得到块，因此它们能够具有对换的东西。除了这5个对等方（“前”4个对等方和一个试探的对等方）的所有其他相邻对等方均被“阻塞”，即它们不能从Alice接收到任何块。BitTorrent有一些有趣的机制没有在这里讨论，包括片（小块）、流水线、随机优先选择、残局模型和反怠慢［Cohen 2003］。

刚刚描述的关于交换的激励机制常被称为“一报还一报”（tit-for-tat）［Cohen 2003］。已证实这种激励方案能被回避［Liogkas 2006；Locher 2006；Piatek 2007］。无论如何，BitTorrent“生态系统”取得了广泛成功，数以百万计的并发对等方在数十万条洪流中积极地共享文件。如果BitTorrent被设计为不采用一报还一报（或一种变种），然而在别的方面却完全相同的协议，BitTorrent现在将很可能不复存在了，因为大多数用户将成为搭便车者了［Sarouiu 2002］。

我们简要地提一下另一种P2P应用——分布式散列表（DHT）来结束我们的讨论。分布式散列表是一种简单的数据库，其数据库记录分布在一个P2P系统的多个对等方上。DHT得到了广泛实现（如在BitTorrent中），并成为大量研究的主题。在配套网站的Video Note中对DHT进行了概述。

2.6 视频流和内容分发网

事先录制的流式视频现在是北美住宅ISP中的流量主体。特别是，Netflix和YouTube在2015年分别消耗了住宅ISP流量的37%和16%［Sandvine 2015］。在本节中，我们将对如何在今天的因特网中实现流行的视频流服务进行概述。我们将看到它们的实现方式是使用应用层协议和以像高速缓存那样方式运行的服务器。在专注于多媒体网络的第9章中，我们将进一步研究因特网视频以及其他因特网多媒体服务。

2.6.1 因特网视频

在流式存储视频应用中，基础的媒体是预先录制的视频，例如电影、电视节目、录制好的体育事件或录制好的用户生成的视频（如通常在YouTube上可见的那些）。这些预先录制好的视频放置在服务器上，用户按需向这些服务器发送请求来观看视频。许多因特网公司现在提供流式视频，这些公司包括Netflix、YouTube（谷歌）、亚马逊和优酷。

但在开始讨论视频流之前，我们先迅速感受一下视频媒体自身。视频是一系列的图像，通常以一种恒定的速率（如每秒24或30张图像）来展现。一幅未压缩、数字编码的图像由像素阵列组成，其中每个像素是由一些比特编码来表示亮度和颜色。视频的一个重要特征是它能够被压缩，因而可用比特率来权衡视频质量。今天现成的压缩算法能够将一个视频压缩成所希望的任何比特率。当然，比特率越高，图像质量越好，用户的总体视觉感受越好。

从网络的观点看，也许视频最为突出的特征是它的高比特率。压缩的因特网视频的比特率范围通常从用于低质量视频的100kbps，到用于流式高分辨率电影的超过3Mbps，再到用于4K流式展望的超过10Mbps。这能够转换为巨大的流量和存储，特别是对高端视频。例如，单一2Mbps视频在67分钟期间将耗费1GB的存储和流量。到目前为止，对流式视频的最为重要的性能度量是平均端到端吞吐量。为了提供连续不断的布局，网络必须为流式应用提供平均吞吐量，这个流式应用至少与压缩视频的比特率一样大。

我们也能使用压缩生成相同视频的多个版本，每个版本有不同的质量等级。例如，我们能够使用压缩生成相同视频的3个版本，比特率分别为300kbps、1Mbps和3Mbps。用户

则能够根据他们当前可用带宽来决定观看哪个版本。具有高速因特网连接的用户也许选择3Mbps版本，使用智能手机通过3G观看视频的用户可能选择300kbps版本。

2.6.2 HTTP流和DASH

在HTTP流中，视频只是存储在HTTP服务器中作为一个普通的文件，每个文件有一个特定的URL。当用户要看该视频时，客户与服务器创建一个TCP连接并发送对该URL的HTTP GET请求。服务器则以底层网络协议和流量条件允许的尽可能快的速率，在一个HTTP响应报文中发送该视频文件。在客户一侧，字节被收集在客户应用缓存中。一旦该缓存中的字节数量超过预先设定的门限，客户应用程序就开始播放，特别是，流式视频应用程序周期性地从客户应用程序缓存中抓取帧，对这些帧解压缩并且在用户屏幕上展现。因此，流式视频应用接收到视频就进行播放，同时缓存该视频后面部分的帧。

如前一小节所述，尽管HTTP流在实践中已经得到广泛部署（例如，自YouTube发展初期开始），但它具有严重缺陷，即所有客户接收到相同编码的视频，尽管对不同的客户或者对于相同客户的不同时间而言，客户可用的带宽大小有很大不同。这导致了一种新型基于HTTP的流的研发，它常常被称为**经HTTP的动态适应性流**（Dynamic Adaptive Streaming over HTTP，DASH）。在DASH中，视频编码为几个不同的版本，其中每个版本具有不同的比特率，对应于不同的质量水平。客户动态地请求来自不同版本且长度为几秒的视频段数据块。当可用带宽量较高时，客户自然地选择来自高速率版本的块；当可用带宽量较低时，客户自然地选择来自低速率版本的块。客户用HTTP GET请求报文一次选择一个不同的块［Akhshabi 2011］。

DASH允许客户使用不同的以太网接入速率流式播放具有不同编码速率的视频。使用低速3G连接的客户能够接收一个低比特率（和低质量）的版本，使用光纤连接的客户能够接收高质量的版本。如果端到端带宽在会话过程中改变的话，DASH允许客户适应可用带宽。这种特色对于移动用户特别重要，当移动用户相对于基站移动时，通常他们能感受到其可用带宽的波动。

使用DASH后，每个视频版本存储在HTTP服务器中，每个版本都有一个不同的URL。HTTP服务器也有一个**告示文件**（manifest file），为每个版本提供了一个URL及其比特率。客户首先请求该告示文件并且得知各种各样的版本。然后客户通过在HTTP GET请求报文中对每块指定一个URL和一个字节范围，一次选择一块。在下载块的同时，客户也测量接收带宽并运行一个*速率决定算法*来选择下次请求的块。自然地，如果客户缓存的视频很多，并且测量的接收带宽较高，它将选择一个高速率的版本。同样，如果客户缓存的视频很少，并且测量的接收带宽较低，它将选择一个低速率的版本。因此DASH允许客户自由地在不同的质量等级之间切换。

2.6.3 内容分发网

今天，许多因特网视频公司日复一日地向数以百万计的用户按需分发每秒数兆比特的流。例如，YouTube的视频库藏有几亿个，每天向全世界的用户分发几亿条流。向位于全世界的所有用户流式传输所有流量同时提供连续播放和高交互性显然是一项有挑战性的任务。

对于一个因特网视频公司，或许提供流式视频服务最为直接的方法是建立单一的大规模数据中心，在数据中心中存储其所有视频，并直接从该数据中心向世界范围的客户传输流式视频。但是这种方法存在三个问题。首先，如果客户远离数据中心，服务器到客户的

分组将跨越许多通信链路并很可能通过许多 ISP，其中某些 ISP 可能位于不同的大洲。如果这些链路之一提供的吞吐量小于视频消耗速率，端到端吞吐量也将小于该消耗速率，给用户带来恼人的停滞时延。（第 1 章讲过，一条流的端到端吞吐量由瓶颈链路的吞吐量所决定。）出现这种事件的可能性随着端到端路径中链路数量的增加而增加。第二个缺陷是流行的视频很可能经过相同的通信链路发送许多次。这不仅浪费了网络带宽，因特网视频公司自己也将为向因特网反复发送相同的字节而向其 ISP 运营商（连接到数据中心）支付费用。这种解决方案的第三个问题是单个数据中心代表一个单点故障，如果数据中心或其通向因特网的链路崩溃，它将不能够分发任何视频流了。

为了应对向分布于全世界的用户分发巨量视频数据的挑战，几乎所有主要的视频流公司都利用**内容分发网**（Content Distribution Network，CDN）。CDN 管理分布在多个地理位置上的服务器，在它的服务器中存储视频（和其他类型的 Web 内容，包括文档、图片和音频）的副本，并且所有试图将每个用户请求定向到一个将提供最好的用户体验的 CDN 位置。CDN 可以是**专用 CDN**（private CDN），即它由内容提供商自己所拥有；例如，谷歌的 CDN 分发 YouTube 视频和其他类型的内容。另一种 CDN 可以是**第三方 CDN**（third-party CDN），它代表多个内容提供商分发内容；Akamai、Limelight 和 Level-3 都运行第三方 CDN。现代 CDN 的一个可读性强的展望见［Leighton 2009；Nygren 2010］。

学习案例

谷歌的网络基础设施

为了支持谷歌的巨量云服务阵列，包括搜索、Gmail、日程表、YouTube 视频、地图、文档和社交网络，谷歌已经部署了一个广泛的专用网和 CDN 基础设施。谷歌的 CDN 基础设施具有三个等级的服务器集群：

- 14 个“百万数据中心”，其中 8 个位于北美，4 个位于欧洲，2 个位于亚洲［Google Locations 2016］，每个数据中心具有 10 万台量级的服务器。这些“百万数据中心”负责服务于动态的（并且经常是个性化的）内容，包括搜索结果和 Gmail 报文。
- 在 IXP 中大约 50 个集群分布于全球，其中每个集群由 100 ~ 500 台服务器组成［Adhikari 2011a］。这些集群负责服务于静态内容，包括 YouTube 视频［Adhikari 2011a］。
- 数以百计的“深入（enter-deep）”集群位于一个接入 ISP 中。这里一个集群通常由位于一个机架上的数十台服务器组成。这些“深入服务器”执行 TCP 分岔（参见 3.7 节）并服务于静态内容［Chen 2011］，包括体现搜索结果的 Web 网页的静态部分。

所有这些数据中心和集群位置与谷歌自己的专用网连接在一起。当某用户进行搜索请求时，该请求常常先经过本地 ISP 发送到邻近的“深入服务器”缓存中，从这里检索静态内容；同时将该静态内容提供给客户，邻近的缓存也经谷歌的专用网将请求转发给“大型数据中心”，从这里检索个性化的搜索结果。对于某 YouTube 视频，该视频本身可能来自一个“邀请坐客服务器”缓存，而围绕该视频的 Web 网页部分可能来自邻近的“深入服务器”缓存，围绕该视频的广告来自数据中心。总的来说，除了本地 ISP，谷歌云服务在很大程度上是由独立于公共因特网的网络基础设施提供的。

CDN 通常采用两种不同的服务器安置原则［Huang 2008］：

- **深入**。第一个原则由 Akamai 首创，该原则是通过在遍及全球的接入 ISP 中部署服务器集群来深入到 ISP 的接入网中。（在 1.3 节中描述了接入网。）Akamai 在大约 1700 个位置采用这种方法部署集群。其目标是靠近端用户，通过减少端用户和 CDN 集群之间（内容从这里收到）链路和路由器的数量，从而改善了用户感受的时延和吞吐量。因为这种高度分布式设计，维护和管理集群的任务成为挑战。
- **邀请做客**。第二个设计原则由 Limelight 和许多其他 CDN 公司所采用，该原则是通过在少量（例如 10 个）关键位置建造大集群来邀请到 ISP 做客。不是将集群放在接入 ISP 中，这些 CDN 通常将它们的集群放置在因特网交换点（IXP）（参见 1.3 节）。与深入设计原则相比，邀请做客设计通常产生较低的维护和管理开销，可能以对端用户的较高时延和较低吞吐量为代价。

一旦 CDN 的集群准备就绪，它就可以跨集群复制内容。CDN 可能不希望将每个视频的副本放置在每个集群中，因为某些视频很少观看或仅在某些国家中流行。事实上，许多 CDN 没有将视频推入它们的集群，而是使用一种简单的拉策略：如果客户向一个未存储该视频的集群请求某视频，则该集群检索该视频（从某中心仓库或者从另一个集群），向客户流式传输视频时的同时在本地存储一个副本。类似于因特网缓存（参见 2.2.5 节），当某集群存储器变满时，它删除不经常请求的视频。

1. CDN 操作

在讨论过这两种部署 CDN 的重要方法后，我们现在深入看看 CDN 操作的细节。当用户主机中的一个浏览器指令检索一个特定的视频（由 URL 标识）时，CDN 必须截获该请求，以便能够：①确定此时适合用于该客户的 CDN 服务器集群；②将客户的请求重定向到该集群的某台服务器。我们很快将讨论 CDN 是如何能够确定一个适当的集群的。但是我们首先考察截获和重定向请求所依赖的机制。

大多数 CDN 利用 DNS 来截获和重定向请求；这种使用 DNS 的一个有趣讨论见［Vixie 2009］。我们考虑用一个简单的例子来说明通常是怎样涉及 DNS 的。假定一个内容提供商 NetCinema，雇佣了第三方 CDN 公司 KingCDN 来向其客户分发视频。在 NetCinema 的 Web 网页上，它的每个视频都被指派了一个 URL，该 URL 包括了字符串“video”以及该视频本身的独特标识符；例如，转换器 7 可以指派为 http://video.netcinema.com/6Y7B23V。接下来出现如图 2-24 所示的 6 个步骤：

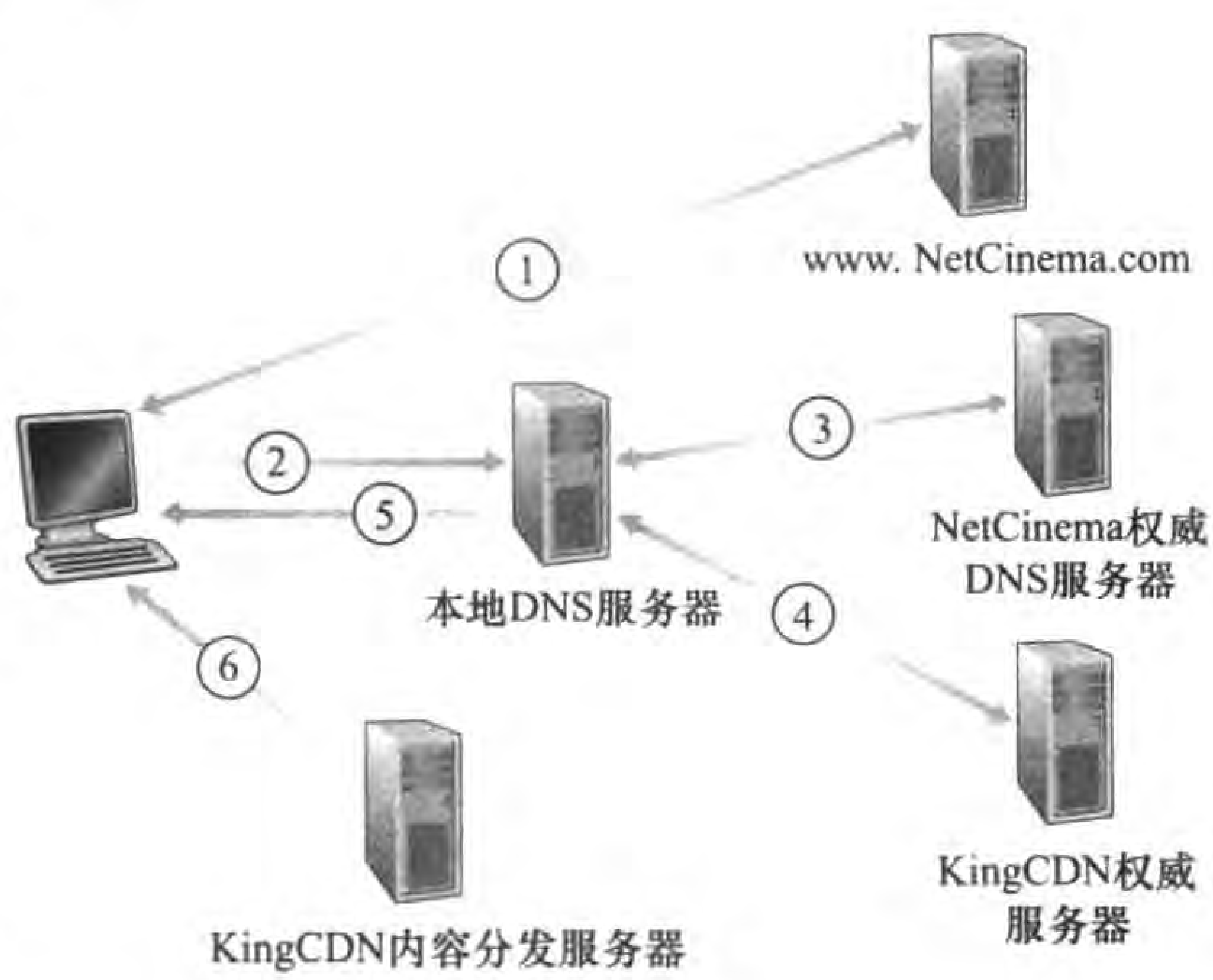


图 2-24　DNS 将用户的请求重定向到一台 CDN 服务器

1）用户访问位于 NetCinema 的 Web 网页。

2）当用户点击链接 http://video.netcinema.com/6Y7B23V 时，该用户主机发送了一个

对于 video. netcinema. com 的 DNS 请求。

3）用户的本地 DNS 服务器（LDNS）将该 DNS 请求中继到一台用于 NetCinema 的权威 DNS 服务器，该服务器观察到主机名 video. netcinema. com 中的字符串“video”。为了将该 DNS 请求移交给 KingCDN，NetCinema 权威 DNS 服务器并不返回一个 IP 地址，而是向 LDNS 返回一个 KingCDN 域的主机名，如 a1105. kingcdn. com。

4）从这时起，DNS 请求进入了 KingCDN 专用 DNS 基础设施。用户的 LDNS 则发送第二个请求，此时是对 a1105. kingcdn. com 的 DNS 请求，KingCDN 的 DNS 系统最终向 LDNS 返回 KingCDN 内容服务器的 IP 地址。所以正是在这里，在 KingCDN 的 DNS 系统中，指定了 CDN 服务器，客户将能够从这台服务器接收到它的内容。

5）LDNS 向用户主机转发内容服务 CDN 节点的 IP 地址。

6）一旦客户收到 KingCDN 内容服务器的 IP 地址，它与具有该 IP 地址的服务器创建了一条直接的 TCP 连接，并且发出对该视频的 HTTP GET 请求。如果使用了 DASH，服务器将首先向客户发送具有 URL 列表的告示文件，每个 URL 对应视频的每个版本，并且客户将动态地选择来自不同版本的块。

2. 集群选择策略

任何 CDN 部署，其核心是**集群选择策略**（cluster selection strategy），即动态地将客户定向到 CDN 中的某个服务器集群或数据中心的机制。如我们刚才所见，经过客户的 DNS 查找，CDN 得知了该客户的 LDNS 服务器的 IP 地址。在得知该 IP 地址之后，CDN 需要基于该 IP 地址选择一个适当的集群。CDN 一般采用专用的集群选择策略。我们现在简单地介绍一些策略，每种策略都有其优点和缺点。

一种简单的策略是指派客户到**地理上最为邻近**（geographically closest）的集群。使用商用地理位置数据库（例如 Quova [Quova 2016] 和 Max-Mind [MaxMind 2016]），每个 LDNS IP 地址都映射到一个地理位置。当从一个特殊的 LDNS 接收到一个 DNS 请求时，CDN 选择地理上最为接近的集群，即离 LDNS 最少几千米远的集群，“就像鸟飞一样”。这样的解决方案对于众多用户来说能够工作得相当好 [Agarwal 2009]。但对于某些客户，该解决方案可能执行的效果差，因为就网络路径的长度或跳数而言，地理最邻近的集群可能并不是最近的集群。此外，一种所有基于 DNS 的方法都内在具有的问题是，某些端用户配置使用位于远地的 LDNS [Shaikh 2001; Mao 2002]，在这种情况下，LDNS 位置可能远离客户的位置。此外，这种简单的策略忽略了时延和可用带宽随因特网路径时间而变化，总是为特定的客户指派相同的集群。

为了基于当前流量条件为客户决定最好的集群，CDN 能够对其集群和客户之间的时延和丢包性能执行周期性的**实时测量**（real-time measurement）。例如，CDN 能够让它的每个集群周期性地向位于全世界的所有 LDNS 发送探测分组（例如，ping 报文或 DNS 请求）。这种方法的一个缺点是许多 LDNS 被配置为不会响应这些探测。

2.6.4 学习案例：Netflix、YouTube 和“看看”

通过观察三个高度成功的大规模部署 Netflix、YouTube 和“看看”，我们来总结对流式存储视频的讨论。我们将看到，这些系统采用的方法差异很大，但却应用了在本节中讨论的许多根本原则。

1. Netflix

Netflix 在 2015 年产生了 37% 的北美住宅 ISP 中的下载流量，它已经成为美国首屈一

指的在线电影和TV节目的服务提供商［Snadvine 2015］。如我们下面讨论的那样，Netflix视频分发具有两个主要部件：亚马逊云和它自己的专用CDN基础设施。

Netflix有一个Web网站来处理若干功能，这些功能包括用户注册和登录、计费、用于浏览和搜索的电影目录以及一个电影推荐系统。如图2-25所示，这个Web网站（以及它关联的后端数据库）完全运行在亚马逊云中的亚马逊服务器上。此外，亚马逊云处理下列关键功能：

- **内容摄取**。在Netflix能够向它的用户分发某电影之前，它必须首先获取和处理该电影。Netflix接收制片厂电影的母带，并且将其上载到亚马逊云的主机上。
- **内容处理**。亚马逊云中的机器为每部电影生成许多不同格式，以适合在桌面计算机、智能手机和与电视机相连的游戏机上运行的不同类型的客户视频播放器。为每种格式和比特率都生成一种不同的版本，允许使用DASH经HTTP适应性播放流。
- **向其CDN上载版本**。一旦某电影的所有版本均已生成，在亚马逊云中的主机向其CDN上载这些版本。

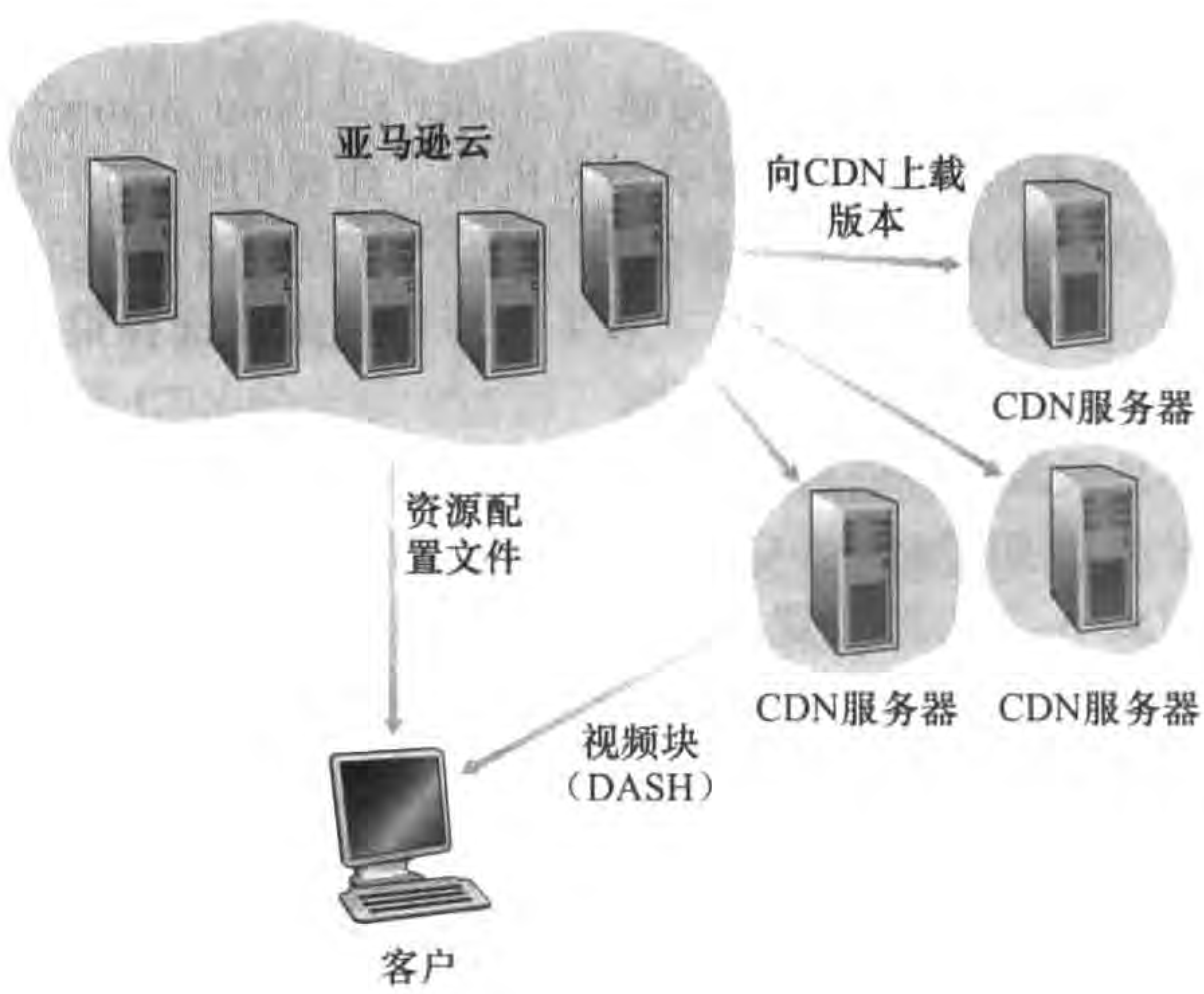


图2-25　Netflix视频流平台

当Netflix于2007年首次推出它的视频流式服务时，它雇佣了3个第三方CDN公司来分发它的视频内容。自那时起，Netflix创建了自己专用的CDN，现在它从这些专用CDN发送它所有的视频。（然而，Netflix仍使用Akamai来分发它的Web网页。）为了创建它自己的CDN，Netflix在IXP和它们自己的住宅ISP中安装了服务器机架。Netflix当前在超过50个IXP位置具有服务器机架；存放Netflix机架的IXP清单可参见［Netflix Open Connect 2016］。也有数百个ISP位置存放Netflix机架，同样可参见［Netflix Open Connect 2016］，其中Netflix向潜在的ISP合作伙伴提供了在其网络中安装一个（免费）Netflix机架的操作指南。在机架中每台服务器具有几个10Gbps以太网端口和超过100TB的存储。在一个机架中服务器的数量是变化的：IXP安装通常有数十台服务器并包含整个Netflix流式视频库（包括多个版本的视频以支持DASH）；本地IXP也许仅有一台服务器并仅包含最为流行的视频。Netflix不使用拉高速缓存（2.2.5节）以在IXP和ISP中扩充它的CDN服务器，反而在非高峰时段通过推将这些视频分发给它的CDN服务器。对于不能保存整个库的那些

位置，Netflix 仅推送最为流行的视频，视频的流行度是基于逐天的数据来决定的。在 YouTube 视频（［Netflix Video 1］和［Netflix Video 2］）中更为详细地描述了 Netflix CDN 设计。

描述了 Netflix 体系结构的组件后，我们来更为仔细地看一下客户与各台服务器之间的交互，这些服务器与电影交付有关。如前面指出的那样，浏览 Netflix 视频库的 Web 网页由亚马逊云中的服务器提供服务。当用户选择一个电影准备播放时，运行在亚马逊云中的 Netflix 软件首先确定它的哪个 CDN 服务器具有该电影的拷贝。在具有拷贝的服务器中，该软件决定客户请求的“最好的”服务器。如果该客户正在使用一个住宅 ISP，它具有安装在该 ISP 中 Netflix CDN 服务器机架并且该机架具有所请求电影的拷贝，则通常选择这个机架中的一台服务器。倘若不是，通常选择邻近 IXP 的一台服务器。

一旦 Netflix 确定了交付内容的 CDN 服务器，它向该客户发送特定服务器的 IP 地址以及资源配置文件，该文件具有所请求电影的不同版本的 URL。该客户和那台 CDN 服务器则使用专用版本的 DASH 进行交互。具体而言，如 2.6.2 节所述，该客户使用 HTTP GET 请求报文中的字节范围首部，以请求来自电影的不同版本的块。Netflix 使用大约 4 秒长的块［Adhikari 2012］。随着这些块的下载，客户测量收到的吞吐量并且运行一个速率确定算法来确定下一个要请求块的质量。

Netflix 包含了本节前面讨论的许多关键原则，包括适应性流和 CDN 分发。然而，因为 Netflix 使用自己专用的 CDN，而它仅分发视频（而非 Web 网页），所以 Netflix 已经能够简化并定制其 CDN 设计。特别是，Netflix 不需要用如 2.6.3 节中所讨论的 DNS 重定向来将特殊的客户连接到一台 CDN 服务器；相反，Netflix 软件（运行在亚马逊云中）直接告知该客户使用一台特定的 CDN 服务器。此外，Netflix CDN 使用推高速缓存而不是拉高速缓存（2.2.5 节）：内容在非高峰时段的预定时间被推入服务器，而不是在高速缓存未命中时动态地被推入。

2. YouTube

YouTube 具有每分钟 300 小时的视频上载和每天几十亿次观看［YouTube 2016］，毫无疑问 YouTube 是世界上最大的视频共享站点。YouTube 于 2005 年 4 月开始它的服务，并于 2006 年 11 月被谷歌公司收购。尽管谷歌/YouTube 设计和协议是专用的，但通过几个独立的测量结果，我们能够基本理解 YouTube 的工作原理［Zink 2009；Torres 2011；Adhikari 2011a］。与 Netflix 一样，YouTube 广泛地利用 CDN 技术来分发它的视频［Torres 2011］。类似于 Netflix，谷歌使用其专用 CDN 来分发 YouTube 视频，并且已经在几百个不同的 IXP 和 ISP 位置安装了服务器集群。从这些位置以及从它的巨大数据中心，谷歌分发 YouTube 视频［Adhikari 2011a］。然而，与 Netflix 不同，谷歌使用如 2.2.5 节中描述的拉高速缓存和如 2.6.3 节中描述的 DNS 重定向。在大部分时间，谷歌的集群选择策略将客户定向到某个集群，使得客户与集群之间的 RTT 是最低的。然而，为了平衡流经集群的负载，有时客户被定向（经 DNS）到一个更远的集群［Torres 2011］。

YouTube 应用 HTTP 流，经常使少量的不同版本为一个视频可用，每个具有不同的比特率和对应的质量等级。YouTube 没有应用适应性流（例如 DASH），而要求用户人工选择一个版本。为了节省那些将被重定位或提前终止而浪费的带宽和服务器资源，YouTube 在获取视频的目标量之后，使用 HTTP 字节范围请求来限制传输的数据流。

每天有几百万视频被上载到 YouTube。不仅 YouTube 视频经 HTTP 以流方式从服务器到客户，而且 YouTube 上载者也经 HTTP 从客户到服务器上载他们的视频。YouTube 处理它收到的每个视频，将它转换为 YouTube 视频格式并且创建具有不同比特率的多个版本。这种处理完全发生在谷歌数据中心。（参见 2.6.3 节中有关谷歌的网络基础设施的学习案例。）

3. 看看

我们刚讨论了由专用 CDN 运行的专用服务器以流的方式向客户发送 Netflix 和 YouTube 视频。Netflix 和 YouTube 不仅必须为服务器硬件付费，而且要为服务器分发视频时使用的带宽付费。考虑到这些服务的规模和它们消耗的带宽量，这种 CDN 部署的代价可能是很高的。

我们通过描述一种完全不同的方法来对本节进行总结，即经因特网大规模地按需提供视频。这是一种允许服务提供商极大地减少其基础设施和带宽成本的方法。如你可能猜测的那样，这种方法使用 P2P 交付而不是客户 - 服务器交付。自 2011 年以来，看看（由迅雷公司拥有并运行）部署的 P2P 视频交付取得了巨大的成功，每个月都有数千万用户 [Zhang 2015]。

从高层面看，P2P 流式视频非常类似于 BitTorrent 文件下载。当一个对等方要看一个视频时，它联系一个跟踪器，以发现在系统中具有该视频副本的其他对等方。这个请求的对等方则并行地从具有该文件的其他对等方请求该视频的块。然而，不同于使用 BitTorrent 下载，请求被优先地给予那些即将播放的块，以确保连续播放 [Dhungel 2012]。

看看近期已经向混合 CDN-P2P 流式系统迁移 [Zhang 2015]。特别是，看看目前在中国部署了数以百计的服务器并且将视频内容推向这些服务器。这个看看 CDN 在流式视频的启动阶段起着主要作用。在大多数场合，客户请求来自 CDN 服务器的内容的开头部分，并且并行地从对等方请求内容。当 P2P 总流量满足视频播放时，该客户将从 CDN 停止流并仅从对等方获得流。但如果 P2P 流的流量不充分，该客户重新启动 CDN 连接并且返回到混合 CDN-P2P 流模式。以这种方式，看看能够确保短启动时延，与此同时最小地依赖成本高的基础设施服务器和带宽。

2.7 套接字编程：生成网络应用

我们已经看到了一些重要的网络应用，下面探讨一下网络应用程序是如何实际编写的。在 2.1 节讲过，典型的网络应用是由一对程序（即客户程序和服务器程序）组成的，它们位于两个不同的端系统中。当运行这两个程序时，创建了一个客户进程和一个服务器进程，同时它们通过从套接字读出和写入数据在彼此之间进行通信。开发者创建一个网络应用时，其主要任务就是编写客户程序和服务器程序的代码。

网络应用程序有两类。一类是由协议标准（如一个 RFC 或某种其他标准文档）中所定义的操作的实现；这样的应用程序有时称为“开放”的，因为定义其操作的这些规则为人们所共知。对于这样的实现，客户程序和服务器程序必须遵守由该 RFC 所规定的规则。例如，某客户程序可能是 HTTP 协议客户端的一种实现，如在 2.2 节所描述，该协议由 RFC 2616 明确定义；类似地，其服务器程序能够是 HTTP 服务器协议的一种实现，也由 RFC 2616 明确定义。如果一个开发者编写客户程序的代码，另一个开发者编写服务器程

序的代码，并且两者都完全遵从该 RFC 的各种规则，那么这两个程序将能够交互操作。实际上，今天许多网络应用程序涉及客户和服务器程序间的通信，这些程序都是由独立的程序员开发的。例如，谷歌 Chrome 浏览器与 Apache Web 服务器通信，BitTorrent 客户与 BitTorrent 跟踪器通信。

另一类网络应用程序是专用的网络应用程序。在这种情况下，由客户和服务器程序应用的应用层协议没有公开发布在某 RFC 中或其他地方。某单独的开发者（或开发团队）产生了客户和服务器程序，并且该开发者用他的代码完全控制该代码的功能。但是因为这些代码并没有实现一个开放的协议，其他独立的开发者将不能开发出和该应用程序交互的代码。

在本节中，我们将考察研发一个客户 - 服务器应用程序中的关键问题，我们将“亲力亲为”来实现一个非常简单的客户 - 服务器应用程序代码。在研发阶段，开发者必须最先做的一个决定是，应用程序是运行在 TCP 上还是运行在 UDP 上。前面讲过 TCP 是面向连接的，并且为两个端系统之间的数据流动提供可靠的字节流通道。UDP 是无连接的，从一个端系统向另一个端系统发送独立的数据分组，不对交付提供任何保证。前面也讲过当客户或服务器程序实现了一个由某 RFC 定义的协议时，它应当使用与该协议关联的周知端口号；与之相反，当研发一个专用应用程序时，研发者必须注意避免使用这些周知端口号。（端口号已在 2.1 节简要讨论过。它们将在第 3 章中更为详细地涉及。）

我们通过一个简单的 UDP 应用程序和一个简单的 TCP 应用程序来介绍 UDP 和 TCP 套接字编程。我们用 Python 3 来呈现这些简单的 TCP 和 UDP 程序。也可以用 Java、C 或 C ++ 来编写这些程序，而我们选择用 Python 最主要原因是 Python 清楚地揭示了关键的套接字概念。使用 Python，代码的行数更少，并且向新编程人员解释每一行代码不会有困难。如果你不熟悉 Python，也用不着担心，只要你有过一些用 Java、C 或 C ++ 编程的经验，就应该很容易看懂下面的代码。

如果读者对用 Java 进行客户 - 服务器编程感兴趣，建议你去查看与本书配套的 Web 网站。事实上，能够在那里找到用 Java 编写的本节中的所有例子（和相关的实验）。如果读者对用 C 进行客户 - 服务器编程感兴趣，有一些优秀参考资料可供使用 [Donahoo 2001; Stevens 1997; Frost 1994; Kurose 1996]。我们下面的 Python 例子具有类似于 C 的外观和感觉。

2.7.1 UDP 套接字编程

在本小节中，我们将编写使用 UDP 的简单客户 - 服务器程序；在下一小节中，我们将编写使用 TCP 的简单程序。

2.1 节讲过，运行在不同机器上的进程彼此通过向套接字发送报文来进行通信。我们说过每个进程好比是一座房子，该进程的套接字则好比是一扇门。应用程序位于房子中门的一侧；运输层位于该门朝外的另一侧。应用程序开发者在套接字的应用层一侧可以控制所有东西；然而，它几乎无法控制运输层一侧。

现在我们仔细观察使用 UDP 套接字的两个通信进程之间的交互。在发送进程能够将数据分组推出套接字之门之前，当使用 UDP 时，必须先将目的地址附在该分组之上。在该分组传过发送方的套接字之后，因特网将使用该目的地址通过因特网为该分组选路到接收进程的套接字。当分组到达接收套接字时，接收进程将通过该套接字取回分组，然后检

查分组的内容并采取适当的动作。

因此你可能现在想知道，附在分组上的目的地址包含了什么？如你所期待的那样，目的主机的 IP 地址是目的地址的一部分。通过在分组中包括目的地的 IP 地址，因特网中的路由器将能够通过因特网将分组选路到目的主机。但是因为一台主机可能运行许多网络应用进程，每个进程具有一个或多个套接字，所以在目的主机指定特定的套接字也是必要的。当生成一个套接字时，就为它分配一个称为**端口号**（port number）的标识符。因此，如你所期待的，分组的目的地址也包括该套接字的端口号。总的来说，发送进程为分组附上目的地址，该目的地址是由目的主机的 IP 地址和目的地套接字的端口号组成的。此外，如我们很快将看到的那样，发送方的源地址也是由源主机的 IP 地址和源套接字的端口号组成，该源地址也要附在分组之上。然而，将源地址附在分组之上通常并不是由 UDP 应用程序代码所为，而是由底层操作系统自动完成的。

我们将使用下列简单的客户－服务器应用程序来演示对于 UDP 和 TCP 的套接字编程：

1）客户从其键盘读取一行字符（数据）并将该数据向服务器发送。

2）服务器接收该数据并将这些字符转换为大写。

3）服务器将修改的数据发送给客户。

4）客户接收修改的数据并在其监视器上将该行显示出来。

图 2-26 着重显示了客户和服务器的主要与套接字相关的活动，两者通过 UDP 运输服务进行通信。

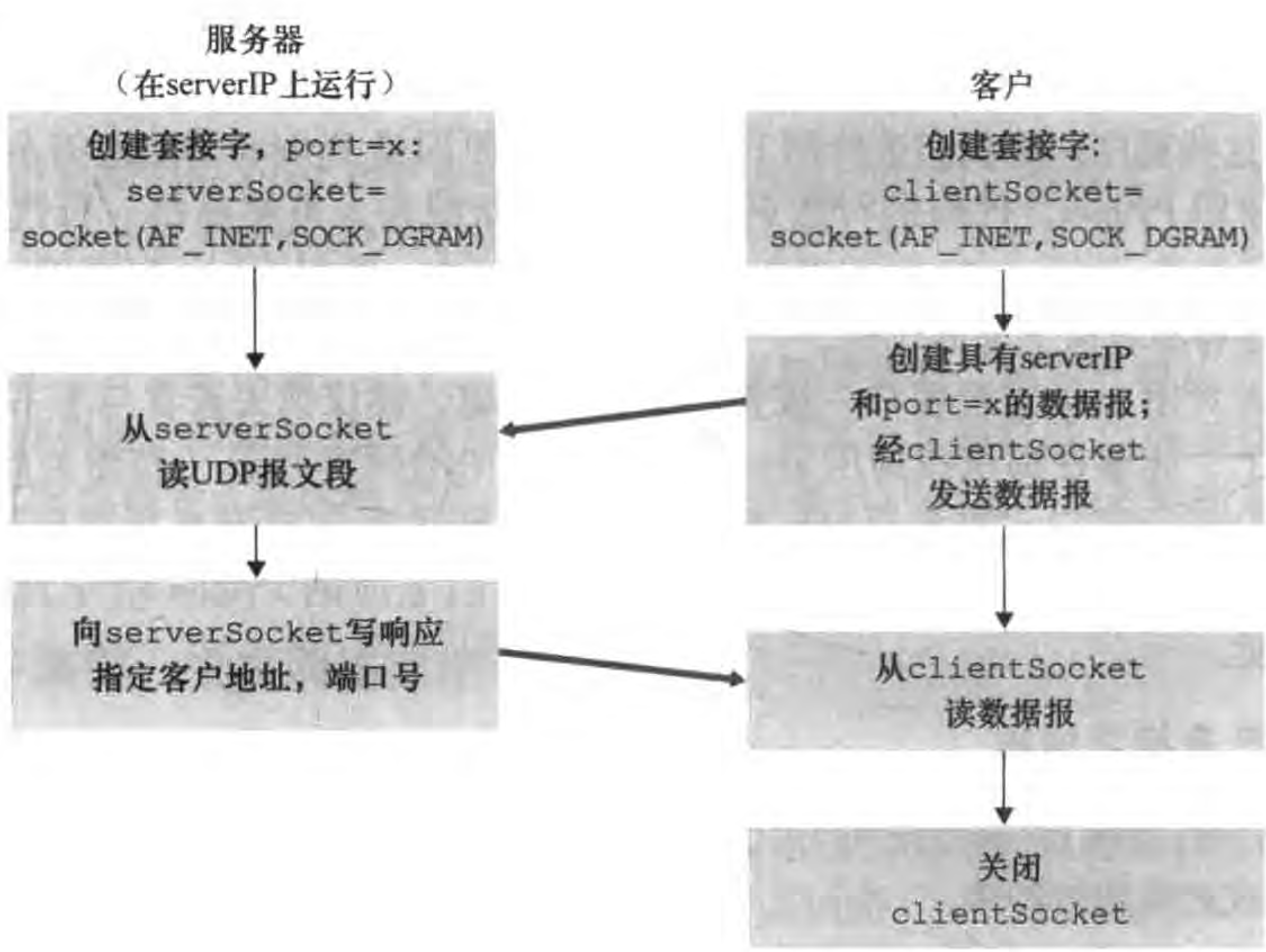


图 2-26　使用 UDP 的客户－服务器应用程序

现在我们自己动手来查看用 UDP 实现这个简单应用程序的一对客户－服务器程序。我们在每个程序后也提供一个详细、逐行的分析。我们将以 UDP 客户开始，该程序将向服务器发送一个简单的应用级报文。服务器为了能够接收并回答该客户的报文，它必须准备好并已经在运行，这就是说，在客户发送其报文之前，服务器必须作为一个进程正在运行。

客户程序被称为 UDPClient. py，服务器程序被称为 UDPServer. py。为了强调关键问题，我们有意提供最少的代码。“好代码”无疑将具有更多辅助性的代码行，特别是用于处理出现差错的情况。对于本应用程序，我们任意选择了 12000 作为服务器的端口号。

1. UDPClient. py

下面是该应用程序客户端的代码：

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message.encode(),(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

现在我们看在 UDPClient. py 中的各行代码。

```
from socket import *
```

该 socket 模块形成了在 Python 中所有网络通信的基础。包括了这行，我们将能够在程序中创建套接字。

```
serverName = 'hostname'
serverPort = 12000
```

第一行将变量 serverName 置为字符串“hostname”。这里，我们提供了或者包含服务器的 IP 地址（如“128. 138. 32. 126”）或者包含服务器的主机名（如“cis. poly. edu”）的字符串。如果我们使用主机名，则将自动执行 DNS lookup 从而得到 IP 地址。第二行将整数变量 serverPort 置为 12000。

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

该行创建了客户的套接字，称为 clientSocket。第一个参数指示了地址簇；特别是，AF_INET 指示了底层网络使用了 IPv4。（此时不必担心，我们将在第 4 章中讨论 IPv4。）第二个参数指示了该套接字是 SOCK_DGRAM 类型的，这意味着它是一个 UDP 套接字（而不是一个 TCP 套接字）。值得注意的是，当创建套接字时，我们并没有指定客户套接字的端口号；相反，我们让操作系统为我们做这件事。既然已经创建了客户进程的门，我们将要生成通过该门发送的报文。

```
message = raw_input('Input lowercase sentence:')
```

raw_input() 是 Python 中的内置功能。当执行这条命令时，客户上的用户将以单词“Input lowercase sentence:”进行提示，用户则使用她的键盘输入一行，该内容被放入变量 message 中。既然我们有了一个套接字和一条报文，我们将要通过该套接字向目的主机发送报文。

```
clientSocket.sendto(message.encode(),(serverName, serverPort))
```

在上述这行中，我们首先将报文由字符串类型转换为字节类型，因为我们需要向套接字中发送字节；这将使用 encode() 方法完成。方法 sendto() 为报文附上目的地址（serverName, serverPort）并且向进程的套接字 clientSocket 发送结果分组。（如前面所述，源地址也附到分组上，尽管这是自动完成的，而不是显式地由代码完成的。）经一

个 UDP 套接字发送一个客户到服务器的报文非常简单！在发送分组之后，客户等待接收来自服务器的数据。

```
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
```

对于上述这行，当一个来自因特网的分组到达该客户套接字时，该分组的数据被放置到变量 modifiedMessage 中，其源地址被放置到变量 serverAddress 中。变量 serverAddress 包含了服务器的 IP 地址和服务器的端口号。程序 UDPClient 实际上并不需要服务器的地址信息，因为它从起始就已经知道了该服务器地址；而这行 Python 代码仍然提供了服务器的地址。方法 recvfrom 也取缓存长度 2048 作为输入。（该缓存长度用于多种目的。）

```
print(modifiedMessage.decode())
```

这行将报文从字节转化为字符串后，在用户显示器上打印出 modifiedMessage。它应当是用户键入的原始行，但现在变为大写的了。

```
clientSocket.close()
```

该行关闭了套接字。然后关闭了该进程。

2. UDPServer. py

现在来看看这个应用程序的服务器端：

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

注意到 UDPServer 的开始部分与 UDPClient 类似。它也是导入套接字模块，也将整数变量 serverPort 设置为 12000，并且也创建套接字类型 SOCK_DGRAM（一种 UDP 套接字）。与 UDPClient 有很大不同的第一行代码是：

```
serverSocket.bind(('', serverPort))
```

上面行将端口号 12000 与该服务器的套接字绑定（即分配）在一起。因此在 UDPServer 中，（由应用程序开发者编写的）代码显式地为该套接字分配一个端口号。以这种方式，当任何人向位于该服务器的 IP 地址的端口 12000 发送一个分组，该分组将导向该套接字。UDPServer 然后进入一个 while 循环；该 while 循环将允许 UDPServer 无限期地接收并处理来自客户的分组。在该 while 循环中，UDPServer 等待一个分组的到达。

```
message, clientAddress = serverSocket.recvfrom(2048)
```

这行代码类似于我们在 UDPClient 中看到的。当某分组到达该服务器的套接字时，该分组的数据被放置到变量 message 中，其源地址被放置到变量 clientAddress 中。变量 clientAddress 包含了客户的 IP 地址和客户的端口号。这里，UDPServer 将利用该地址信息，因为它提供了返回地址，类似于普通邮政邮件的返回地址。使用该源地址信息，服务器此时知道了它应当将回答发向何处。

```
modifiedMessage = message.decode().upper()
```

此行是这个简单应用程序的关键部分。它在将报文转化为字符串后，获取由客户发送的行并使用方法 upper()将其转换为大写。

```
serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

最后一行将该客户的地址（IP 地址和端口号）附到大写的报文上（在将字符串转化为字节后），并将所得的分组发送到服务器的套接字中。（如前面所述，服务器地址也附在分组上，尽管这是自动而不是显式地由代码完成的。）然后因特网将分组交付到该客户地址。在服务器发送该分组后，它仍维持在 while 循环中，等待（从运行在任一台主机上的任何客户发送的）另一个 UDP 分组到达。

为了测试这对程序，可在一台主机上运行 UDPClient. py，并在另一台主机上运行 UDPServer. py。保证在 UDPClient. py 中包括适当的服务器主机名或 IP 地址。接下来，在服务器主机上执行编译的服务器程序 UDPServer. py。这在服务器上创建了一个进程，等待着某个客户与之联系。然后，在客户主机上执行编译的客户器程序 UDPClient. py。这在客户上创建了一个进程。最后，在客户上使用应用程序，键入一个句子并以回车结束。

可以通过稍加修改上述客户和服务器程序来研制自己的 UDP 客户 - 服务器程序。例如，不必将所有字母转换为大写，服务器可以计算字母 s 出现的次数并返回该数字。或者能够修改客户程序，使其在收到一个大写的句子后，用户能够向服务器继续发送更多的句子。

2.7.2 TCP 套接字编程

与 UDP 不同，TCP 是一个面向连接的协议。这意味着在客户和服务器能够开始互相发送数据之前，它们先要握手和创建一个 TCP 连接。TCP 连接的一端与客户套接字相联系，另一端与服务器套接字相联系。当创建该 TCP 连接时，我们将其与客户套接字地址（IP 地址和端口号）和服务器套接字地址（IP 地址和端口号）关联起来。使用创建的 TCP 连接，当一侧要向另一侧发送数据时，它只需经过其套接字将数据丢进 TCP 连接。这与 UDP 不同，UDP 服务器在将分组丢进套接字之前必须为其附上一个目的地地址。

现在我们仔细观察一下 TCP 中客户程序和服务器程序的交互。客户具有向服务器发起接触的任务。服务器为了能够对客户的初始接触做出反应，服务器必须已经准备好。这意味着两件事。第一，与在 UDP 中的情况一样，TCP 服务器在客户试图发起接触前必须作为进程运行起来。第二，服务器程序必须具有一扇特殊的门，更精确地说是一个特殊的套接字，该门欢迎来自运行在任意主机上的客户进程的某种初始接触。使用房子与门来比喻进程与套接字，有时我们将客户的初始接触称为"敲欢迎之门"。

随着服务器进程的运行，客户进程能够向服务器发起一个 TCP 连接。这是由客户程序通过创建一个 TCP 套接字完成的。当该客户生成其 TCP 套接字时，它指定了服务器中的欢迎套接字的地址，即服务器主机的 IP 地址及其套接字的端口号。生成其套接字后，该客户发起了一个三次握手并创建与服务器的一个 TCP 连接。发生在运输层的三次握手，对于客户和服务器程序是完全透明的。

在三次握手期间，客户进程敲服务器进程的欢迎之门。当该服务器"听"到敲门声时，它将生成一扇新门（更精确地讲是一个*新*套接字），它专门用于特定的客户。在我们下面的例子中，欢迎之门是一个我们称为 serverSocket 的 TCP 套接字对象；它是专门对客

户进行连接的新生成的套接字，称为连接套接字（connectionSocket）。初次遇到 TCP 套接字的学生有时会混淆欢迎套接字（这是所有要与服务器通信的客户的起始接触点）和每个新生成的服务器侧的连接套接字（这是随后为与每个客户通信而生成的套接字）。

从应用程序的观点来看，客户套接字和服务器连接套接字直接通过一根管道连接。如图 2-27 所示，客户进程可以向它的套接字发送任意字节，并且 TCP 保证服务器进程能够按发送的顺序接收（通过连接套接字）每个字节。TCP 因此在客户和服务器进程之间提供了可靠服务。此外，就像人们可以从同一扇门进和出一样，客户进程不仅能向它的套接字发送字节，也能从中接收字节；类似地，服务器进程不仅从它的连接套接字接收字节，也能向其发送字节。

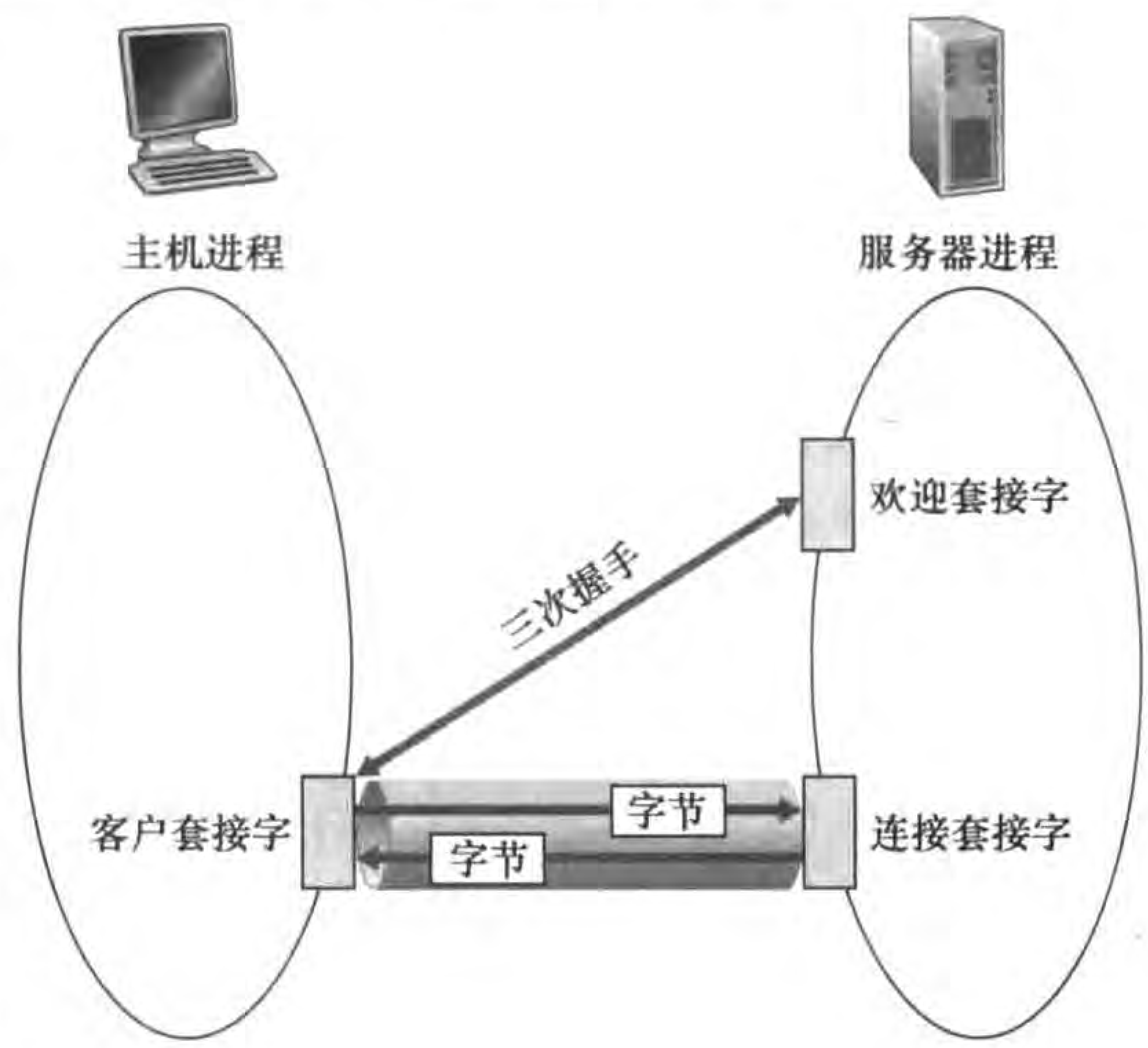


图 2-27　TCPServer 进程有两个套接字

我们使用同样简单的客户 - 服务器应用程序来展示 TCP 套接字编程：客户向服务器发送一行数据，服务器将这行改为大写并回送给客户。图 2-28 着重显示了客户和服务器的主要与套接字相关的活动，两者通过 TCP 运输服务进行通信。

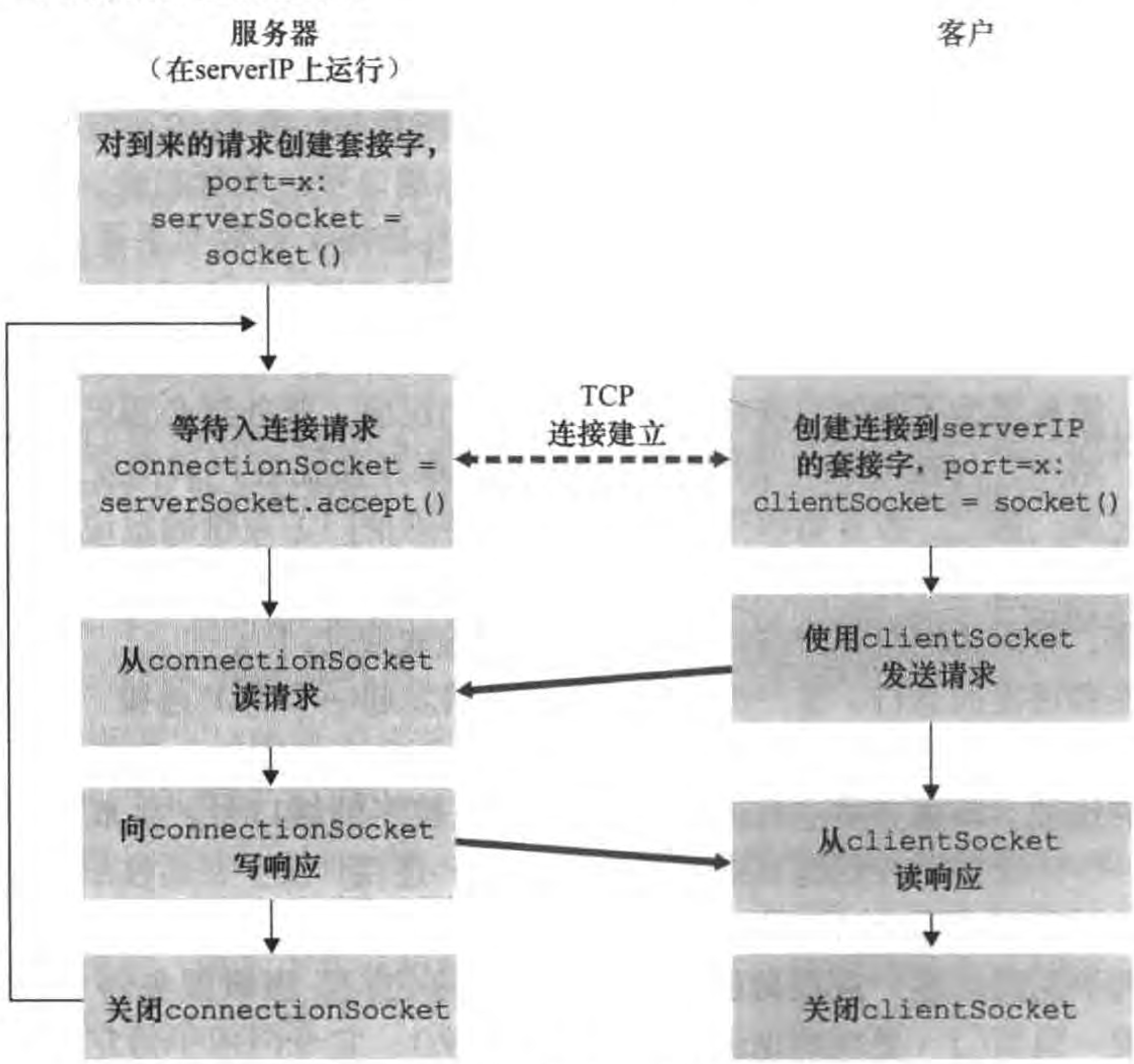


图 2-28　使用 TCP 的客户 - 服务器应用程序

1. TCPClient. py

这里给出了应用程序客户端的代码：

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server: ', modifiedSentence.decode())
clientSocket.close()
```

现在我们查看这些代码中与 UDP 实现有很大差别的各行。首当其冲的行是客户套接字的创建。

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

该行创建了客户的套接字，称为 clientSocket。第一个参数仍指示底层网络使用的是 IPv4。第二个参数指示该套接字是 SOCK_STREAM 类型。这表明它是一个 TCP 套接字（而不是一个 UDP 套接字）。值得注意的是当我们创建该客户套接字时仍未指定其端口号；相反，我们让操作系统为我们做此事。此时的下一行代码与我们在 UDPClient 中看到的极为不同：

```
clientSocket.connect((serverName,serverPort))
```

前面讲过在客户能够使用一个 TCP 套接字向服务器发送数据之前（反之亦然），必须在客户与服务器之间创建一个 TCP 连接。上面这行就发起了客户和服务器之间的这条 TCP 连接。connect() 方法的参数是这条连接中服务器端的地址。这行代码执行完后，执行三次握手，并在客户和服务器之间创建起一条 TCP 连接。

```
sentence = raw_input('Input lowercase sentence:')
```

如同 UDPClient 一样，上一行从用户获得了一个句子。字符串 sentence 连续收集字符直到用户键入回车以终止该行为止。代码的下一行也与 UDPClient 极为不同：

```
clientSocket.send(sentence.encode())
```

上一行通过该客户的套接字并进入 TCP 连接发送字符串 sentence。值得注意的是，该程序并未显式地创建一个分组并为该分组附上目的地址，而使用 UDP 套接字却要那样做。相反，该客户程序只是将字符串 sentence 中的字节放入该 TCP 连接中去。客户然后就等待接收来自服务器的字节。

```
modifiedSentence = clientSocket.recv(2048)
```

当字符到达服务器时，它们被放置在字符串 modifiedSentence 中。字符继续积累在 modifiedSentence 中，直到该行以回车符结束为止。在打印大写句子后，我们关闭客户的套接字。

```
clientSocket.close()
```

最后一行关闭了套接字，因此关闭了客户和服务器之间的 TCP 连接。它引起客户中的 TCP 向服务器中的 TCP 发送一条 TCP 报文（参见 3.5 节）。

2. TCPServer. py

现在我们看一下服务器程序。

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print('The server is ready to receive')
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.encode())
    connectionSocket.close()
```

现在我们来看看上述与 UDPServer 及 TCPClient 有显著不同的代码行。与 TCPClient 相同的是，服务器创建一个 TCP 套接字，执行：

```
serverSocket=socket(AF_INET,SOCK_STREAM)
```

与 UDPServer 类似，我们将服务器的端口号 serverPort 与该套接字关联起来：

```
serverSocket.bind(('',serverPort))
```

但对 TCP 而言，serverSocket 将是我们的欢迎套接字。在创建这扇欢迎之门后，我们将等待并聆听某个客户敲门：

```
serverSocket.listen(1)
```

该行让服务器聆听来自客户的 TCP 连接请求。其中参数定义了请求连接的最大数（至少为 1）。

```
connectionSocket, addr = serverSocket.accept()
```

当客户敲该门时，程序为 serverSocket 调用 accept()方法，这在服务器中创建了一个称为 connectionSocket 的新套接字，由这个特定的客户专用。客户和服务器则完成了握手，在客户的 clientSocket 和服务器的 serverSocket 之间创建了一个 TCP 连接。借助于创建的 TCP 连接，客户与服务器现在能够通过该连接相互发送字节。使用 TCP，从一侧发送的所有字节不仅确保到达另一侧，而且确保按序到达。

```
connectionSocket.close()
```

在此程序中，在向客户发送修改的句子后，我们关闭了该连接套接字。但由于 serverSocket 保持打开，所以另一个客户此时能够敲门并向该服务器发送一个句子要求修改。

我们现在完成了 TCP 套接字编程的讨论。建议你在两台单独的主机上运行这两个程序，也可以修改它们以达到稍微不同的目的。你应当将前面两个 UDP 程序与这两个 TCP 程序进行比较，观察它们的不同之处。你也应当做在第 2、4 和 9 章后面描述的套接字编程作业。最后，我们希望在掌握了这些和更先进的套接字程序后的某天，你将能够编写你自己的流行网络应用程序，变得非常富有和声名卓著，并记得本书的作者！

2.8　小结

在本章中，我们学习了网络应用的概念和实现两个方面。我们学习了被因特网应用普遍采用的客户-服务器模式，并且看到了该模式在 HTTP、SMTP、POP3 和 DNS 等协议中的使用。我们已经更为详细地学习了这些重要的应用层协议以及与之对应的相关应用（Web、文件传输、电子邮件和 DNS）。我们也已学习了 P2P 体系结构以及它如何应用在许多应用程序中。我们也学习了流式视频，以及现代视频分发系统是如何利用 CDN 的。对

于面向连接的（TCP）和无连接的（UDP）端到端传输服务，我们走马观花般地学习了套接字的使用。至此，我们在分层的网络体系结构中的向下之旅已经完成了第一步。

在本书一开始的 1.1 节中，我们对协议给出了一个相当含糊的框架性定义："在两个或多个通信实体之间交换报文的格式和次序，以及对某报文或其他事件传输和/或接收所采取的动作。"本章中的内容，特别是我们对 HTTP、SMTP、POP3 和 DNS 协议进行的细致研究，已经为这个定义加入了相当可观的实质性的内容。协议是网络连接中的核心概念；对应用层协议的学习，为我们提供了有关协议内涵的更为直觉的认识。

在 2.1 节中，我们描述了 TCP 和 UDP 为调用它们的应用提供的服务模型。当我们在 2.7 节中开发运行在 TCP 和 UDP 之上的简单应用程序时，我们对这些服务模型进行了更加深入的观察。然而，我们几乎没有介绍 TCP 和 UDP 是如何提供这种服务模型的。例如，我们知道 TCP 提供了一种可靠数据服务，但我们未说它是如何做到这一点的。在下一章中我们将不仅关注运输协议是什么，而且还关注它*如何*工作以及*为什么*要这么做。

有了因特网应用程序结构和应用层协议的知识之后，我们现在准备继续沿该协议栈向下，在第 3 章中探讨运输层。

课后习题和问题

复习题

2.1 节

R1. 列出 5 种非专用的因特网应用及它们所使用的应用层协议。

R2. 网络体系结构与应用程序体系结构之间有什么区别？

R3. 对两进程之间的通信会话而言，哪个进程是客户，哪个进程是服务器？

R4. 对一个 P2P 文件共享应用，你同意"一个通信会话不存在客户端和服务器端的概念"的说法吗？为什么？

R5. 运行在一台主机上的一个进程，使用什么信息来标识运行在另一台主机上的进程？

R6. 假定你想尽快地处理从远程客户到服务器的事务，你将使用 UDP 还是 TCP？为什么？

R7. 参见图 2-4，我们看到在该图中所列出的应用程序没有一个同时既要求无数据丢失又要求定时的。你能设想一个既要求无数据丢失又高度时间敏感的应用程序吗？

R8. 列出一个运输协议能够提供的 4 种宽泛类型的服务。对于每种服务类型，指出是 UDP 还是 TCP（或这两种协议）提供这样的服务？

R9. 前面讲过 TCP 能用 SSL 来强化，以提供进程到进程的安全性服务，包括加密。SSL 运行在运输层还是应用层？如果某应用程序研制者想要用 SSL 来强化 UDP，该研制者应当做些什么工作？

2.2 ~ 2.4 节

R10. 握手协议的作用是什么？

R11. 为什么 HTTP、SMTP 及 POP3 都运行在 TCP，而不是 UDP 上？

R12. 考虑一个电子商务网站需要保留每一个客户的购买记录。描述如何使用 cookie 来完成该功能？

R13. 描述 Web 缓存器是如何减少接收被请求对象的时延的。Web 缓存器将减少一个用户请求的所有对象或只是其中的某些对象的时延吗？为什么？

R14. Telnet 到一台 Web 服务器并发送一个多行的请求报文。在该请求报文中包含 If - modified - since：首部行，迫使响应报文中出现"304 Not Modified"状态代码。

R15. 列出几种流行的即时通信应用。它们使用相同的协议作为 SMS 吗？

R16. 假定 Alice 使用一个基于 Web 的电子邮件账户（例如 Hotmail 或 Gmail）向 Bob 发报文，而 Bob 使用

POP3 从他的邮件服务器访问自己的邮件。讨论该报文是如何从 Alice 主机到 Bob 主机的。要列出在两台主机间移动该报文时所使用的各种应用层协议。

R17. 将你最近收到的报文首部打印出来。其中有多少 Received：首部行？分析该报文的首部行中的每一行。

R18. 从用户的观点看，POP3 协议中下载并删除模式和下载并保留模式有什么区别吗？

R19. 一个机构的 Web 服务器和邮件服务器可以有完全相同的主机名别名（例如，foo.com）吗？包含邮件服务器主机名的 RR 有什么样的类型？

R20. 仔细检查收到的电子邮件，查找由使用.edu 电子邮件地址的用户发送的报文首部。从其首部，能够确定发送该报文的主机的 IP 地址吗？对于由 Gmail 账号发送的报文做相同的事。

2.5 节

R21. 在 BitTorrent 中，假定 Alice 向 Bob 提供一个 30 秒间隔的文件块吞吐量。Bob 将必须进行回报，在相同的间隔中向 Alice 提供文件块吗？为什么？

R22. 考虑一个新对等方 Alice 加入 BitTorrent 而不拥有任何文件块。没有任何块，因此她没有任何东西可上载，她无法成为任何其他对等方的前 4 位上载者。那么 Alice 将怎样得到她的第一个文件块呢？

R23. 覆盖网络是什么？它包括路由器吗？在覆盖网络中边是什么？

2.6 节

R24. CDN 通常采用两种不同的服务器放置方法之一。列举并简单描述它们。

R25. 除了如时延、丢包和带宽性能等网络相关的考虑外，设计一种 CDN 服务器选择策略时还有其他重要因素。它们是什么？

2.7 节

R26. 2.7 节中所描述的 UDP 服务器仅需要一个套接字，而 TCP 服务器需要两个套接字。为什么？如果 TCP 服务器支持 n 个并行连接，每条连接来自不同的客户主机，那么 TCP 服务器将需要多少个套接字？

R27. 对于 2.7 节所描述的运行在 TCP 之上的客户 - 服务器应用程序，服务器程序为什么必须先于客户程序运行？对于运行在 UDP 之上的客户 - 服务器应用程序，客户程序为什么可以先于服务器程序运行？

习题

P1. 是非判断题。

a. 假设用户请求由一些文本和 3 幅图像组成的 Web 页面。对于这个页面，客户将发送一个请求报文并接收 4 个响应报文。

b. 两个不同的 Web 页面（例如，www.mit.edu/research.html 及 www.mit.edu/students.html）可以通过同一个持续连接发送。

c. 在浏览器和初始服务器之间使用非持续连接的话，一个 TCP 报文段是可能携带两个不同的 HTTP 服务请求报文的。

d. 在 HTTP 响应报文中的 Date：首部指出了该响应中对象最后一次修改的时间。

e. HTTP 响应报文决不会具有空的报文体。

P2. SMS、iMessage 和 WhatsApp 都是智能手机即时通信系统。在因特网上进行一些研究后，为这些系统分别写一段它们所使用协议的文字。然后撰文解释它们的差异所在。

P3. 考虑一个要获取给定 URL 的 Web 文档的 HTTP 客户。该 HTTP 服务器的 IP 地址开始时并不知道。在这种情况下，除了 HTTP 外，还需要什么运输层和应用层协议？

P4. 考虑当浏览器发送一个 HTTP GET 报文时，通过 Wireshark 俘获到下列 ASCII 字符串（即这是一个 HTTP GET 报文的实际内容）。字符 <*cr*> <*lf*> 是回车和换行符（即下面文本中的斜体字符串 <*cr*> 表示了单个回车符，该回车符包含在 HTTP 首部中的相应位置）。回答下列问题，指出你在下面 HTTP GET 报文中找到答案的地方。

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml, application/xml, application/xhtml+xml, text
/html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-
Encoding: zip,deflate<cr><lf>Accept-Charset: ISO
-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

a. 由浏览器请求的文档的 URL 是什么?

b. 该浏览器运行的是 HTTP 的何种版本?

c. 该浏览器请求的是一条非持续连接还是一条持续连接?

d. 该浏览器所运行的主机的 IP 地址是什么?

e. 发起该报文的浏览器的类型是什么? 在一个 HTTP 请求报文中，为什么需要浏览器类型?

P5. 下面文本中显示的是来自服务器的回答，以响应上述问题中 HTTP GET 报文。回答下列问题，指出你在下面报文中找到答案的地方。

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46
GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf>Accept-
Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>
Keep-Alive: timeout=max=100<cr><lf>Connection:
Keep-Alive<cr><lf>Content-Type: text/html; charset=
ISO-8859-1<cr><lf><cr><lf><!doctype html public "-
//w3c//dtd html 4.0transitional//en"><lf><html><lf>
<head><lf> <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1"><lf> <meta
name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT
5.0; U) Netscape]"><lf> <title>CMPSCI 453 / 591 /
NTU-ST550ASpring 2005 homepage</title><lf></head><lf>
<much more document text following here (not shown)>
```

a. 服务器能否成功地找到那个文档? 该文档提供回答是什么时间?

b. 该文档最后修改是什么时间?

c. 文档中被返回的字节有多少?

d. 文档被返回的前 5 个字节是什么? 该服务器同意一条持续连接吗?

P6. 获取 HTTP/1.1 规范（RFC 2616）。回答下面问题:

a. 解释在客户和服务器之间用于指示关闭持续连接的信令机制。客户、服务器或两者都能发送信令通知连接关闭吗?

b. HTTP 提供了什么加密服务?

c. 一个客户能够与一个给定的服务器打开 3 条或更多条并发连接吗?

d. 如果一个服务器或一个客户检测到连接已经空闲一段时间，该服务器或客户可以关闭两者之间的传输连接。一侧开始关闭连接而另一侧通过该连接传输数据是可能的吗? 请解释。

P7. 假定你在浏览器中点击一条超链接获得 Web 页面。相关联的 URL 的 IP 地址没有缓存在本地主机上，因此必须使用 DNS lookup 以获得该 IP 地址。如果主机从 DNS 得到 IP 地址之前已经访问了 n 个 DNS 服务器；相继产生的 RTT 依次为 RTT_1、…、RTT_n。进一步假定与链路相关的 Web 页面只包含一个对象，即由少量的 HTML 文本组成。令 RTT_0 表示本地主机和包含对象的服务器之间的 RTT 值。假定该对象传输时间为零，则从该客户点击该超链接到它接收到该对象需要多长时间?

P8. 参照习题 P7，假定在同一服务器上某 HTML 文件引用了 8 个非常小的对象。忽略发送时间，在下列情况下需要多长时间:

a. 没有并行 TCP 连接的非持续 HTTP。

b. 配置有 5 个并行连接的非持续 HTTP。

c. 持续 HTTP。

P9. 考虑图 2-12，其中有一个机构的网络和因特网相连。假定对象的平均长度为 850 000 比特，从这个机构网的浏览器到初始服务器的平均请求率是每秒 16 个请求。还假定从接入链路的因特网一侧的路由器转发一个 HTTP 请求开始，到接收到其响应的平均时间是 3 秒（参见 2.2.5 节）。将总的平均响应时间建模为平均接入时延（即从因特网路由器到机构路由器的时延）和平均因特网时延之和。对于平均接入时延，使用 $\Delta/(1-\Delta\beta)$，式中 Δ 是跨越接入链路发送一个对象的平均时间，β 是对象对该接入链路的平均到达率。

a. 求出总的平均响应时间。

b. 现在假定在这个机构 LAN 中安装了一个缓存器。假定命中率为 0.4，求出总的响应时间。

P10. 考虑一条 10 米短链路，某发送方经过它能够以 150bps 速率双向传输。假定包含数据的分组是 100 000比特长，仅包含控制（如 ACK 或握手）的分组是 200 比特长。假定 N 个并行连接每个都获得 $1/N$ 的链路带宽。现在考虑 HTTP 协议，并且假定每个下载对象是 100Kb 长，这些初始下载对象包含 10 个来自相同发送方的引用对象。在这种情况下，经非持续 HTTP 的并行实例的并行下载有意义吗？现在考虑持续 HTTP。你期待这比非持续的情况有很大增益吗？评价并解释你的答案。

P11. 考虑在前一个习题中引出的情况。现在假定该链路由 Bob 和 4 个其他用户所共享。Bob 使用非持续 HTTP 的并行实例，而其他 4 个用户使用无并行下载的非持续 HTTP。

a. Bob 的并行连接能够帮助他更快地得到 Web 页面吗？

b. 如果所有 5 个用户打开 5 个非持续 HTTP 并行实例，那么 Bob 的并行连接仍将是有好处的吗？为什么？

P12. 写一个简单的 TCP 程序，使服务器接收来自客户的行并将其打印在服务器的标准输出上。（可以通过修改本书中的 TCPServer. py 程序实现上述任务。）编译并执行你的程序。在另一台有浏览器的机器上，设置浏览器的代理服务器为你正在运行服务器程序的机器，同时适当地配置端口号。这时你的浏览器向服务器发送 GET 请求报文，你的服务器应当在其标准输出上显示该报文。使用这个平台来确定你的浏览器是否对本地缓存的对象产生了条件 GET 报文。

P13. SMTP 中的 MAIL FROM 与该邮件报文自身中的 From：之间有什么不同？

P14. SMTP 是怎样标识一个报文体结束的？HTTP 是怎样做的呢？HTTP 能够使用与 SMTP 标识一个报文体结束相同的方法吗？试解释。

P15. 阅读用于 SMTP 的 RFC 5321。MTA 代表什么？考虑下面收到的垃圾邮件（从一份真实垃圾邮件修改得到）。假定这封垃圾邮件的唯一始作俑者是恶意的，而其他主机是诚实的，指出产生了这封垃圾邮件的恶意主机。

```
From - Fri Nov 07 13:41:30 2008
Return-Path: <tennis5@pp33head.com>
Received: from barmail.cs.umass.edu (barmail.cs.umass.
edu
[128.119.240.3]) by cs.umass.edu (8.13.1/8.12.6) for
<hg@cs.umass.edu>; Fri, 7 Nov 2008 13:27:10 -0500
Received: from asusus-4b96 (localhost [127.0.0.1]) by
barmail.cs.umass.edu (Spam Firewall) for <hg@cs.umass.
edu>; Fri, 7
Nov 2008 13:27:07 -0500 (EST)
Received: from asusus-4b96 ([58.88.21.177]) by barmail.
cs.umass.edu
for <hg@cs.umass.edu>; Fri, 07 Nov 2008 13:27:07 -0500
(EST)
Received: from [58.88.21.177] by inbnd55.exchangeddd.
com; Sat, 8
Nov 2008 01:27:07 +0700
From: "Jonny" <tennis5@pp33head.com>
To: <hg@cs.umass.edu>

Subject: How to secure your savings
```

P16. 阅读 POP3 的 RFC，即 RFC 1939。UIDL POP3 命令的目的是什么？

P17. 考虑用 POP3 访问你的电子邮件。

a. 假定你已经配置以下载并删除模式运行的 POP 邮件客户。完成下列事务：

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: ..........blah
S: .
?
?
```

b. 假定你已经配置以下载并保持模式运行的 POP 邮件客户。完成下列事务：

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: ..........blah
S: .
?
?
```

c. 假定你已经配置以下载并保持模式运行的 POP 邮件客户。使用（b）中的记录，假定你检索报文 1 和 2，退出 POP，5 分钟以后，你再访问 POP 以检索新电子邮件。假定在这 5 分钟间隔内，没有新报文发送给你。给出第二种 POP 会话的记录。

P18. 如题：

a. 什么是 whois 数据库？

b. 使用因特网上的各种 whois 数据库，获得两台 DNS 服务器的名字。指出你使用的是哪个 whois 数据库。

c. 你本地机器上使用 nslookup 向 3 台 DNS 服务器发送 DNS 查询：你的本地 DNS 服务器和两台你在（b）中发现的 DNS 服务器。尝试对类型 A、NS 和 MX 报告进行查询。总结你的发现。

d. 使用 nslookup 找出一台具有多个 IP 地址的 Web 服务器。你所在的机构（学校或公司）的 Web 服务器具有多个 IP 地址吗？

e. 使用 ARIN whois 数据库，确定你所在大学使用的 IP 地址范围。

f. 描述一个攻击者在发动攻击前，能够怎样利用 whois 数据库和 nslookup 工具来执行对一个机构的侦察。

g. 讨论为什么 whois 数据库应当为公众所用。

P19. 在本习题中，我们使用在 Unix 和 Linux 主机上可用的 dig 工具来探索 DNS 服务器的等级结构。图 2-18讲过，在 DNS 等级结构中较高的 DNS 服务器授权对该等级结构中较低 DNS 服务器的 DNS 请求，这是通过向 DNS 客户发送回那台较低层次的 DNS 服务器的名字来实现的。先阅读 dig 的帮助页，再回答下列问题。

a. 从一台根 DNS 服务器（从根服务器 [a-m].root-server.net 之一）开始，通过使用 dig 得到你所在系的 Web 服务器的 IP 地址，发起一系列查询。显示回答你的查询的授权链中的 DNS 服务器的名字列表。

b. 对几个流行 Web 站点如 google. com、yahoo. com 或 amazon. com，重复上一小题。

P20. 假定你能够访问所在系的本地 DNS 服务器中的缓存。你能够提出一种方法来粗略地确定在你所在系的用户中最为流行的 Web 服务器（你所在系以外）吗？解释原因。

P21. 假设你所在系具有一台用于系里所有计算机的本地 DNS 服务器。你是普通用户（即你不是网络/系统管理员）。你能够确定是否在几秒前从你系里的一台计算机可能访问过一台外部 Web 站点吗？解释原因。

P22. 考虑向 N 个对等方分发 $F = 15\text{Gb}$ 的一个文件。该服务器具有 $u_s = 30\text{Mbps}$ 的上载速率，每个对等方具有 $d_i = 2\text{Mbps}$ 的下载速率和上载速率 u。对于 $N = 10$、100 和 1000 并且 $u = 300\text{kbps}$、700kbps 和

2Mbps，对于 N 和 u 的每种组合绘制出确定最小分发时间的图表。需要分别针对客户－服务器分发和 P2P 分发两种情况制作。

P23. 考虑使用一种客户－服务器体系结构向 N 个对等方分发一个 F 比特的文件。假定一种某服务器能够同时向多个对等方传输的流体模型，只要组合速率不超过 u_s，则以不同的速率向每个对等方传输。

a. 假定 $u_s/N \leqslant d_{min}$。定义一个具有 NF/u_s 分发时间的分发方案。

b. 假定 $u_s/N \geqslant d_{min}$。定义一个具有 F/d_{min} 分发时间的分发方案。

c. 得出最小分发时间通常是由 $\max\{NF/u_s, F/d_{min}\}$ 所决定的结论。

P24. 考虑使用 P2P 体系结构向 N 个用户分发 F 比特的一个文件。假定一种流体模型。为了简化起见，假定 d_{min} 很大，因此对等方下载带宽不会成为瓶颈。

a. 假定 $u_s \leqslant (u_s + u_1 + \cdots + u_N)/N$。定义一个具有 F/u_s 分发时间的分发方案。

b. 假定 $u_s \geqslant (u_s + u_1 + \cdots + u_N)/N$。定义一个具有 $NF/(u_s + u_1 + \cdots + u_N)$ 分发时间的分发方案。

c. 得出最小分发时间通常是由 $\max\{F/u_s, NF/(u_s + u_1 + \cdots + u_N)\}$ 所决定的结论。

P25. 考虑在一个有 N 个活跃对等方的覆盖网络中，每对对等方有一条活跃的 TCP 连接。此外，假定该 TCP 连接通过总共 M 台路由器。在对应的覆盖网络中，有多少节点和边？

P26. 假定 Bob 加入 BitTorrent，但他不希望向任何其他对等方上载任何数据（因此称为搭便车）。

a. Bob 声称他能够收到由该社区共享的某文件的完整副本。Bob 所言是可能的吗？为什么？

b. Bob 进一步声称他还能够更为有效地进行他的“搭便车”，方法是利用所在系的计算机实验室中的多台计算机（具有不同的 IP 地址）。他怎样才能做到这些呢？

P27. 考虑一个具有 N 个视频版本（具有 N 个不同的速率和质量）和 N 个音频版本（具有 N 个不同的速率和质量）的 DASH 系统。假设我们想允许播放者在任何时间选择 N 个视频版本和 N 个音频版本之一：

a. 如果我们生成音频与视频混合的文件，因此服务器在任何时间仅发送一个媒体流，该服务器将需要存储多少个文件（每个文件有一个不同的 URL）？

b. 如果该服务器分别发送音频流和视频流并且与客户同步这些流，该服务器将需要存储多少个文件？

P28. 在一台主机上安装编译 TCPClient 和 UDPClient Python 程序，在另一台主机上安装编译 TCPServer 和 UDPServer 程序。

a. 假设你在运行 TCPServer 之前运行 TCPClient，将发生什么现象？为什么？

b. 假设你在运行 UDPServer 之前运行 UDPClient，将发生什么现象？为什么？

c. 如果你对客户端和服务器端使用了不同的端口，将发生什么现象？

P29. 假定在 UDPClient. py 中在创建套接字后增加了下面一行：

```
clientSocket.bind(('', 5432))
```

有必要修改 UDPServer. py 吗？UDPClient 和 UDPServer 中的套接字端口号是多少？在变化之前它们是多少？

P30. 你能够配置浏览器以打开对某 Web 站点的多个并行连接吗？有大量的并行 TCP 连接的优点和缺点是什么？

P31. 我们已经看到因特网 TCP 套接字将数据处理为字节流，而 UDP 套接字识别报文边界。面向字节 API 与显式识别和维护应用程序定义的报文边界的 API 相比，试给出一个优点和一个缺点。

P32. 什么是 Apache Web 服务器？它值多少钱？它当前有多少功能？为回答这个问题，你也许要看一下维基百科。

套接字编程作业

配套 Web 网站包括了 6 个套接字编程作业。前四个作业简述如下。第 5 个作业利用了 ICMP 协议，在第 5 章结尾简述。第 6 个作业使用了多媒体协议，在第 9 章结尾进行总结。极力推荐学生完成这些作业中的几个（如果不是全部的话）。学生能够在 Web 网站 http://www.pearsonhighered.com/cs-resources 上找到这些作业的全面细节，以及 Python 代码的重要片段。

作业1：Web服务器

在这个编程作业中，你将用Python语言开发一个简单的Web服务器，它仅能处理一个请求。具体而言，你的Web服务器将：(1) 当一个客户（浏览器）联系时创建一个连接套接字；(2) 从这个连接接收HTTP请求；(3) 解释该请求以确定所请求的特定文件；(4) 从服务器的文件系统获得请求的文件；(5) 创建一个由请求的文件组成的HTTP响应报文，报文前面有首部行；(6) 经TCP连接向请求的浏览器发送响应。如果浏览器请求一个在该服务器中不存在的文件，服务器应当返回一个"404 Not Found"差错报文。

在配套网站中，我们提供了用于该服务器的框架代码。你的任务是完善该代码，运行你的服务器，通过在不同主机上运行的浏览器发送请求来测试该服务器。如果运行你服务器的主机上已经有一个Web服务器在运行，你应当为该Web服务器使用一个不同于80端口的其他端口。

作业2：UDP ping程序

在这个编程作业中，你将用Python编写一个客户ping程序。该客户将发送一个简单的ping报文，接收一个从服务器返回的对应pong报文，并确定从该客户发送ping报文到接收到pong报文为止的时延。该时延称为往返时延（RTT）。由该客户和服务器提供的功能类似于在现代操作系统中可用的标准ping程序。然而，标准的ping使用互联网控制报文协议（ICMP）（我们将在第5章中学习ICMP）。此时我们将创建一个非标准（但简单）的基于UDP的ping程序。

你的ping程序经UDP向目标服务器发送10个ping报文。对于每个报文，当对应的pong报文返回时，你的客户要确定和打印RTT。因为UDP是一个不可靠的协议，由客户发送的分组可能会丢失。为此，客户不能无限期地等待对ping报文的回答。客户等待服务器回答的时间至多为1秒；如果没有收到回答，客户假定该分组丢失并相应地打印一条报文。

在此作业中，你将给出服务器的完整代码（在配套网站中可找到）。你的任务是编写客户代码，该代码与服务器代码非常类似。建议你先仔细学习服务器的代码，然后编写你的客户代码，可以随意地从服务器代码中剪贴代码行。

作业3：邮件客户

这个编程作业的目的是创建一个向任何接收方发送电子邮件的简单邮件客户。你的客户将必须与邮件服务器（如谷歌的电子邮件服务器）创建一个TCP连接，使用SMTP协议与邮件服务器进行交谈，经该邮件服务器向某接收方（如你的朋友）发送一个电子邮件报文，最后关闭与该邮件服务器的TCP连接。

对本作业，配套Web站点为你的客户提供了框架代码。你的任务是完善该代码并通过向不同的用户账户发送电子邮件来测试你的客户。你也可以尝试通过不同的服务器（例如谷歌的邮件服务器和你所在大学的邮件服务器）进行发送。

作业4：多线程Web代理服务器

在这个编程作业中，你将研发一个简单的Web代理服务器。当你的代理服务器从一个浏览器接收到对某对象的HTTP请求，它生成对相同对象的一个新HTTP请求并向初始服务器发送。当该代理从初始服务器接收到具有该对象的HTTP响应时，它生成一个包括该对象的新HTTP响应，并发送给该客户。这个代理将是多线程的，使其在相同时间能够处理多个请求。

对本作业而言，配套Web网站对该代理服务器提供了框架代码。你的任务是完善该代码，然后测试你的代理，方法是让不同的浏览器经过你的代理来请求Web对象。

Wireshark实验：HTTP

在实验1中，我们已经初步使用了Wireshark分组嗅探器，现在准备使用Wireshark来研究运行中的协议。在本实验中，我们将研究HTTP协议的几个方面：基本的GET/回答交互，HTTP报文格式，检索大HTML文件，检索具有内嵌URL的HTML文件，持续和非持续连接，HTTP鉴别和安全性。

如同所有的Wireshark实验一样，对该实验的全面描述可查阅本书的Web站点http://www.pearsonhighered.com/cs-resources。

Wireshark 实验：DNS

在本实验中，我们仔细观察 DNS 的客户端（DNS 是用于将因特网主机名转换为 IP 地址的协议）。2.4 节讲过，在 DNS 中客户角色是相当简单的：客户向它的本地 DNS 服务器发送一个请求，并接收返回的响应。在此过程中发生的很多事情均不为 DNS 客户所见，如等级结构的 DNS 服务器互相通信递归地或迭代地解析该客户的 DNS 请求。然而，从 DNS 客户的角度而言，该协议是相当简单的，即向本地 DNS 服务器发送一个请求，从该服务器接收一个响应。在本实验中我们观察运转中的 DNS。

如同所有的 Wireshark 实验一样，在本书的 Web 站点 http://www.awl.com/kurose-ross 可以找到本实验的完整描述。

人物专访

Marc Andreessen 是 Mosaic 的共同发明人，Mosaic 是一种 Web 浏览器，正是它使万维网在 1993 年流行起来。Mosaic 具有一个清晰、易于理解的界面，是首个能嵌在文本中显示图像的浏览器。在 1994 年，Marc Andreessen 和 Jim Clark 创办了 Netscape 公司，其浏览器是到 20 世纪 90 年代中期为止最为流行的。Netscape 也研发了安全套接字层（SSL）协议和许多因特网服务器产品，包括邮件服务器和基于 SSL 的 Web 服务器。他现在是风险投资公司 Andreessen Horowitz 的共同奠基人和一般股东，对包括 Facebook、Foursquare、Groupon、Jawbone、Twitter 和 Zynga 等公司的财产投资搭配进行监管。他服务于包括 Bump、eBay、Glam Media、Facebook 和 HP 等在内的多个董事会。他具有美国伊利诺伊大学厄巴纳-香槟分校的计算机科学理学学士学位。


Marc Andreessen

- 您是怎样变得对计算感兴趣的？您过去一直知道您要从事信息技术吗？

在我长大成人的过程中，视频游戏和个人计算正好成为成功而风行一时的事物，在 20 世纪 70 年代后期和 80 年代初期，个人计算成为新技术发展前沿。那时不只有苹果和 IBM 的个人计算机，而且有如 Commodore 和 Atari 等数以百计的新公司。我在 10 岁时用一本名为《简明 BASIC 速成》（Instant Freeze-Dried BASIC）的书进行自学，并在 12 岁时得到自己的第一台计算机（TRS-80 Color Computer，查查它！）。

- 请描述您职业生涯中干过的一两个最令人激动的项目。最大的挑战是什么？

毋庸置疑，最令人兴奋的项目是 1992～1993 年的初始 Mosaic Web 浏览器，最大的挑战是让任何人从此往后都认真地对待它。在那个时候，每个人都认为交互式未来将是由大型公司宣布的“交互式电视”，而非由新兴公司发明的因特网。

- 您对网络和因特网未来的什么东西感到兴奋？您最为关注什么？

最为兴奋的东西是程序员和企业家能够探索的巨大的尚待开发的应用和服务领域，即因特网已经释放的创造性到达了一种我们以前从未预见到的水平。我最关注是“意想不到的后果”的原则，即我们并不总是知道我们所做事情的后果，例如因特网被政府所用，使监视居民到达了一种新水平。

- 随着 Web 技术的进展，学生们有什么应当特别要了解的？

改变的速度，即对学习来说，最重要的东西是学习的方法，在特定的技术中如何灵活地适应改变，当你在职业生涯中前行时，在新的机会和可能性方面如何保持开放的思想。

- 是谁激发了您的职业灵感？

他们是：Vannevar Bush，Ted Nelson，Doug Engelgart，Nolan Bushnell，Bill Hewlett 和 Dave Packard，Ken Olsen，Steve Jobs，Steve Wozniak，Andy Grove，Grace Hopper，Hedy Lamarr，Alan Turing，Richard Stallman。

- 对于要在计算和信息技术领域谋求发展的学生，您有什么忠告？

尽可能深入地理解技术是怎样创造的，然后补充学习商业运作的原理。

- 技术能够解决世界的问题吗？

不能，但是通过经济增长我们推动人们生活标准的改善。综观历史，大多数经济增长来自技术，因此就像技术带来的好处一样。

运 输 层

运输层位于应用层和网络层之间，是分层的网络体系结构的重要部分。该层为运行在不同主机上的应用进程提供直接的通信服务起着至关重要的作用。我们在本章采用的教学方法是，交替地讨论运输层的原理和这些原理在现有的协议中是如何实现的。与往常一样，我们将特别关注因特网协议，即 TCP 和 UDP 运输层协议。

我们将从讨论运输层和网络层的关系开始。这就为研究运输层第一个关键功能打好了基础，即将网络层的在两个端系统之间的交付服务扩展到运行在两个不同端系统上的应用层进程之间的交付服务。我们将在讨论因特网的无连接运输协议 UDP 时阐述这个功能。

然后我们重新回到原理学习上，面对计算机网络中最为基础性的问题之一，即两个实体怎样才能在一种会丢失或损坏数据的媒体上可靠地通信。通过一系列复杂性不断增加（从而更真实！）的场景，我们将逐步建立起一套被运输协议用来解决这些问题的技术。然后，我们将说明这些原理是如何体现在因特网面向连接的运输协议 TCP 中的。

接下来我们讨论网络中的第二个基础性的重要问题，即控制运输层实体的传输速率以避免网络中的拥塞，或从拥塞中恢复过来。我们将考虑拥塞的原因和后果，以及常用的拥塞控制技术。在透彻地理解了拥塞控制问题之后，我们将研究 TCP 应对拥塞控制的方法。

3.1 概述和运输层服务

在前两章中，我们已对运输层的作用及其所提供的服务有所了解。现在我们快速地回顾一下前面学过的有关运输层的知识。

运输层协议为运行在不同主机上的应用进程之间提供了**逻辑通信**（logic communication）功能。从应用程序的角度看，通过逻辑通信，运行不同进程的主机好像直接相连一样；实际上，这些主机也许位于地球的两侧，通过很多路由器及多种不同类型的链路相连。应用进程使用运输层提供的逻辑通信功能彼此发送报文，而无须考虑承载这些报文的物理基础设施的细节。图 3-1 图示了逻辑通信的概念。

如图 3-1 所示，运输层协议是在端系统中而不是在路由器中实现的。在发送端，运输层将从发送应用程序进程接收到的报文转换成运输层分组，用因特网术语来讲该分组称为运输层**报文段**（segment）。实现的方法（可能）是将应用报文划分为较小的块，并为每块加上一个运输层首部以生成运输层报文段。然后，在发送端系统中，运输层将这些报文段传递给网络层，网路层将其封装成网络层分组（即数据报）并向目的地发送。注意到下列事实是重要的：网络路由器仅作用于该数据报的网络层字段；即它们不检查封装在该数据报的运输层报文段的字段。在接收端，网络层从数据报中提取运输层报文段，并将该报文段向上交给运输层。运输层则处理接收到的报文段，使该报文段中的数据为接收应用进程使用。

网络应用程序可以使用多种的运输层协议。例如，因特网有两种协议，即 TCP 和 UDP。每种协议都能为调用的应用程序提供一组不同的运输层服务。

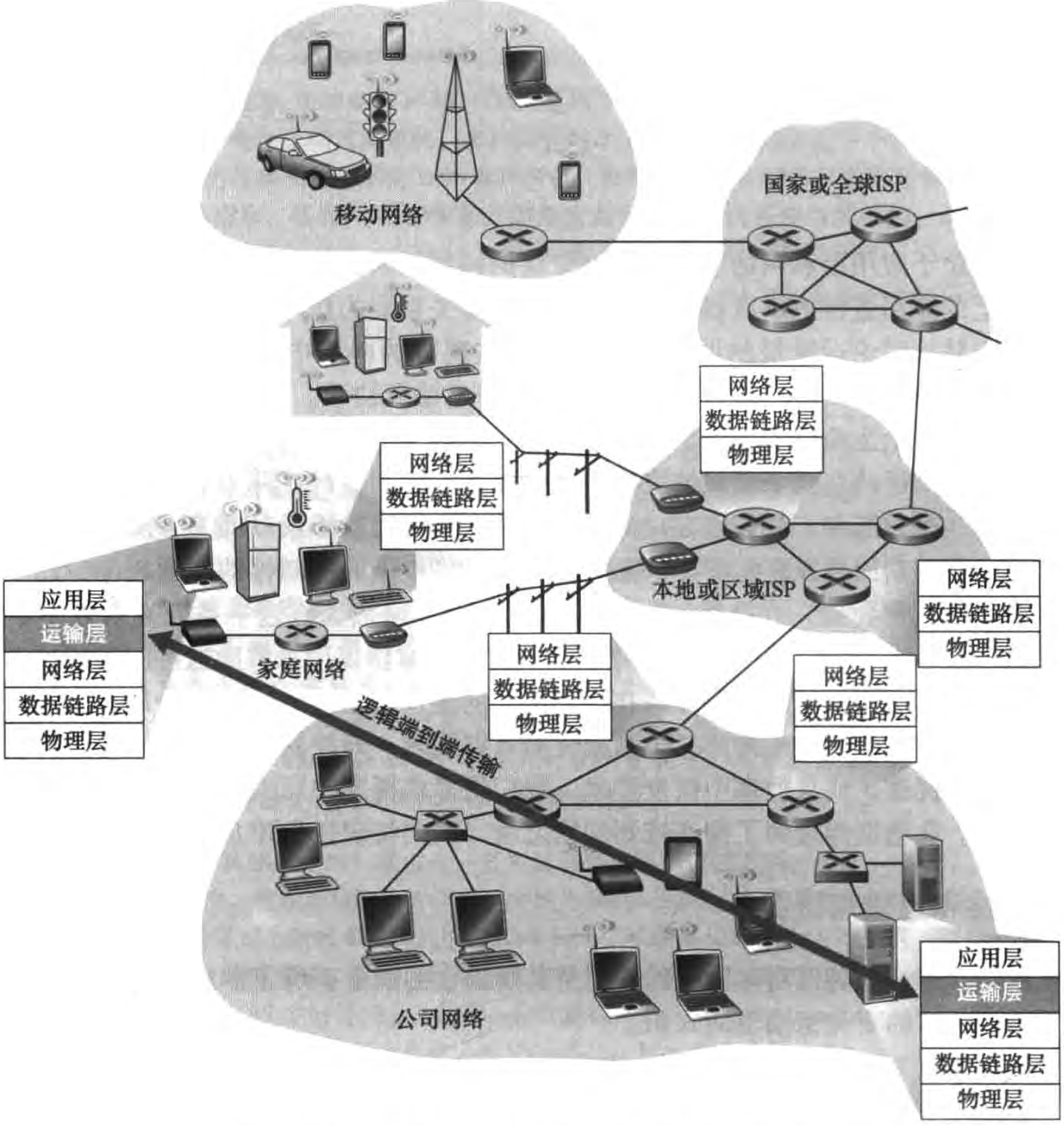


图 3-1 运输层在应用程序进程间提供逻辑的而非物理的通信

3.1.1 运输层和网络层的关系

前面讲过，在协议栈中，运输层刚好位于网络层之上。网络层提供了主机之间的逻辑通信，而运输层为运行在不同主机上的进程之间提供了逻辑通信。这种差别虽然细微但很重要。我们用一个家庭类比来帮助分析这种差别。

考虑有两个家庭，一家位于美国东海岸，一家位于美国西海岸，每家有 12 个孩子。东海岸家庭的孩子们是西海岸家庭孩子们的堂兄弟姐妹。这两个家庭的孩子们喜欢彼此通信，每个人每星期要互相写一封信，每封信都用单独的信封通过传统的邮政服务传送。因此，每个家庭每星期向另一家发送 144 封信。（如果他们有电子邮件的话，这些孩子可以省不少钱！）每一个家庭有个孩子负责收发邮件，西海岸家庭是 Ann 而东海岸家庭是 Bill。每星期 Ann 去她的所有兄弟姐妹那里收集信件，并将这些信件交到每天到家门口来的邮政服务的邮车上。当信件到达西海岸家庭时，Ann 也负责将信件分发到她的兄弟姐妹手上。在东海岸家庭中的 Bill 也负责类似的工作。

在这个例子中，邮政服务为两个家庭间提供逻辑通信，邮政服务将信件从一家送往另一家，而不是从一个人送往另一个人。在另一方面，Ann 和 Bill 为堂兄弟姐妹之间提供了逻辑通信，Ann 和 Bill 从兄弟姐妹那里收取信件或到兄弟姐妹那里交付信件。注意到从堂兄弟姐妹们的角度来看，Ann 和 Bill 就是邮件服务，尽管他们只是端到端交付过程的一部分（即端系统部分）。在解释运输层和网络层之间的关系时，这个家庭的例子是一个非常好的类比。

应用层报文 = 信封上的字符
进程 = 堂兄弟姐妹
主机(又称为端系统) = 家庭
运输层协议 = Ann 和 Bill
网络层协议 = 邮政服务(包括邮车)

我们继续观察这个类比。值得注意的是，Ann 和 Bill 都是在各自家里进行工作的；例如，他们并没有参与任何一个中间邮件中心对邮件进行分拣，或者将邮件从一个邮件中心送到另一个邮件中心之类的工作。类似地，运输层协议只工作在端系统中。在端系统中，运输层协议将来自应用进程的报文移动到网络边缘（即网络层），反过来也是一样，但对有关这些报文在网络核心如何移动并不作任何规定。事实上，如图 3-1 所示，中间路由器既不处理也不识别运输层加在应用层报文的任何信息。

我们还是继续讨论这两家的情况。现在假定 Ann 和 Bill 外出度假，另外一对堂兄妹（如 Susan 和 Harvey）接替他们的工作，在家庭内部进行信件的收集和交付工作。不幸的是，Susan 和 Harvey 的收集和交付工作与 Ann 和 Bill 所做的并不完全一样。由于年龄更小，Susan 和 Harvey 收发邮件的次数更少，而且偶尔还会丢失邮件（有时是被家里的狗咬坏了）。因此，Susan 和 Harvey 这对堂兄妹并没有提供与 Ann 和 Bill 一样的服务集合（即相同的服务模型）。与此类似，计算机网络中可以安排多种运输层协议，每种协议为应用程序提供不同的服务模型。

Ann 和 Bill 所能提供的服务明显受制于邮政服务所能提供的服务。例如，如果邮政服务不能提供在两家之间传递邮件所需时间的最长期限（例如 3 天），那么 Ann 和 Bill 就不可能保证邮件在堂兄弟姐妹之间传递信件的最长期限。与此类似，运输协议能够提供的服务常常受制于底层网络层协议的服务模型。如果网络层协议无法为主机之间发送的运输层报文段提供时延或带宽保证的话，运输层协议也就无法为进程之间发送的应用程序报文提供时延或带宽保证。

然而，即使底层网络协议不能在网络层提供相应的服务，运输层协议也能提供某些服务。例如，如我们将在本章所见，即使底层网络协议是不可靠的，也就是说网络层协议会使分组丢失、篡改和冗余，运输协议也能为应用程序提供可靠的数据传输服务。另一个例子是（我们在第 8 章讨论网络安全时将会研究到），即使网络层不能保证运输层报文段的机密性，运输协议也能使用加密来确保应用程序报文不被入侵者读取。

3.1.2 因特网运输层概述

前面讲过因特网为应用层提供了两种截然不同的可用运输层协议。这些协议一种是 UDP（用户数据报协议），它为调用它的应用程序提供了一种不可靠、无连接的服务。另一种是 TCP（传输控制协议），它为调用它的应用程序提供了一种可靠的、面向连接的服务。当设计一个网络应用程序时，该应用程序的开发人员必须指定使用这两种运输协议中

的哪一种。如我们在2.7节看到的那样，应用程序开发人员在生成套接字时必须指定是选择UDP还是选择TCP。

为了简化术语，我们将运输层分组称为*报文段*（segment）。然而，因特网文献（如RFC文档）也将TCP的运输层分组称为报文段，而常将UDP的分组称为数据报（datagram）。而这类因特网文献也将网络层分组称为*数据报*！本书作为一本计算机网络的入门书籍，我们认为将TCP和UDP的分组统称为*报文段*，而将*数据报*名称保留给网络层分组不容易混淆。

在对UDP和TCP进行简要介绍之前，简单介绍一下因特网的网络层（我们将在第4和5章中详细地学习网络层）是有用的。因特网网络层协议有一个名字叫IP，即网际协议。IP为主机之间提供了逻辑通信。IP的服务模型是**尽力而为交付服务**（best-effort delivery service）。这意味着IP尽它“最大的努力”在通信的主机之间交付报文段，*但它并不做任何确保*。特别是，它不确保报文段的交付，不保证报文段的按序交付，不保证报文段中数据的完整性。由于这些原因，IP被称为**不可靠服务**（unreliable service）。在此还要指出的是，每台主机至少有一个网络层地址，即所谓的IP地址。我们在第4和5章将详细讨论IP地址；在这一章中，我们只需要记住*每台主机有一个IP地址*。

在对IP服务模型有了初步了解后，我们总结一下UDP和TCP所提供的服务模型。UDP和TCP最基本的责任是，将两个端系统间IP的交付服务扩展为运行在端系统上的两个进程之间的交付服务。将主机间交付扩展到进程间交付被称为**运输层的多路复用**（transport-layer multiplexing）与**多路分解**（demultiplexing）。我们将在下一节讨论运输层的多路复用与多路分解。UDP和TCP还可以通过在其报文段首部中包括差错检查字段而提供完整性检查。进程到进程的数据交付和差错检查是两种最低限度的运输层服务，也是UDP所能提供的仅有的两种服务。特别是，与IP一样，UDP也是一种不可靠的服务，即不能保证一个进程所发送的数据能够完整无缺地（或全部！）到达目的进程。在3.3节中将更详细地讨论UDP。

另一方面，TCP为应用程序提供了几种附加服务。首先，它提供**可靠数据传输**（reliable data transfer）。通过使用流量控制、序号、确认和定时器（本章将详细介绍这些技术），TCP确保正确地、按序地将数据从发送进程交付给接收进程。这样，TCP就将两个端系统间的不可靠IP服务转换成了一种进程间的可靠数据传输服务。TCP还提供**拥塞控制**（congestion control）。拥塞控制与其说是一种提供给调用它的应用程序的服务，不如说是一种提供给整个因特网的服务，这是一种带来通用好处的服务。不太严格地说，TCP拥塞控制防止任何一条TCP连接用过多流量来淹没通信主机之间的链路和交换设备。TCP力求为每个通过一条拥塞网络链路的连接平等地共享网络链路带宽。这可以通过调节TCP连接的发送端发送进网络的流量速率来做到。在另一方面，UDP流量是不可调节的。使用UDP传输的应用程序可以根据其需要以其愿意的任何速率发送数据。

一个能提供可靠数据传输和拥塞控制的协议必定是复杂的。我们将用几节的篇幅来介绍可靠数据传输和拥塞控制的原理，用另外几节介绍TCP协议本身。3.4～3.8节将研究这些主题。本章采取基本原理和TCP协议交替介绍的方法。例如，我们首先在一般环境下讨论可靠数据传输，然后讨论TCP是怎样具体提供可靠数据传输的。类似地，先在一般环境下讨论拥塞控制，然后讨论TCP是怎样实现拥塞控制的。但在全面介绍这些内容之前，我们先学习运输层的多路复用与多路分解。

3.2 多路复用与多路分解

在本节中，我们讨论运输层的多路复用与多路分解，也就是将由网络层提供的主机到主机交付服务延伸到为运行在主机上的应用程序提供进程到进程的交付服务。为了使讨论具体起见，我们将在因特网环境中讨论这种基本的运输层服务。然而，需要强调的是，多路复用与多路分解服务是所有计算机网络都需要的。

在目的主机，运输层从紧邻其下的网络层接收报文段。运输层负责将这些报文段中的数据交付给在主机上运行的适当应用程序进程。我们来看一个例子。假定你正坐在计算机前下载 Web 页面，同时还在运行一个 FTP 会话和两个 Telnet 会话。这样你就有 4 个网络应用进程在运行，即两个 Telnet 进程，一个 FTP 进程和一个 HTTP 进程。当你的计算机中的运输层从底层的网络层接收数据时，它需要将所接收到的数据定向到这 4 个进程中的一个。现在我们来研究这是怎样完成的。

首先回想 2.7 节的内容，一个进程（作为网络应用的一部分）有一个或多个**套接字**（socket），它相当于从网络向进程传递数据和从进程向网络传递数据的门户。因此，如图 3-2 所示，在接收主机中的运输层实际上并没有直接将数据交付给进程，而是将数据交给了一个中间的套接字。由于在任一时刻，在接收主机上可能有不止一个套接字，所以每个套接字都有唯一的标识符。标识符的格式取决于它是 UDP 还是 TCP 套接字，我们将很快对它们进行讨论。

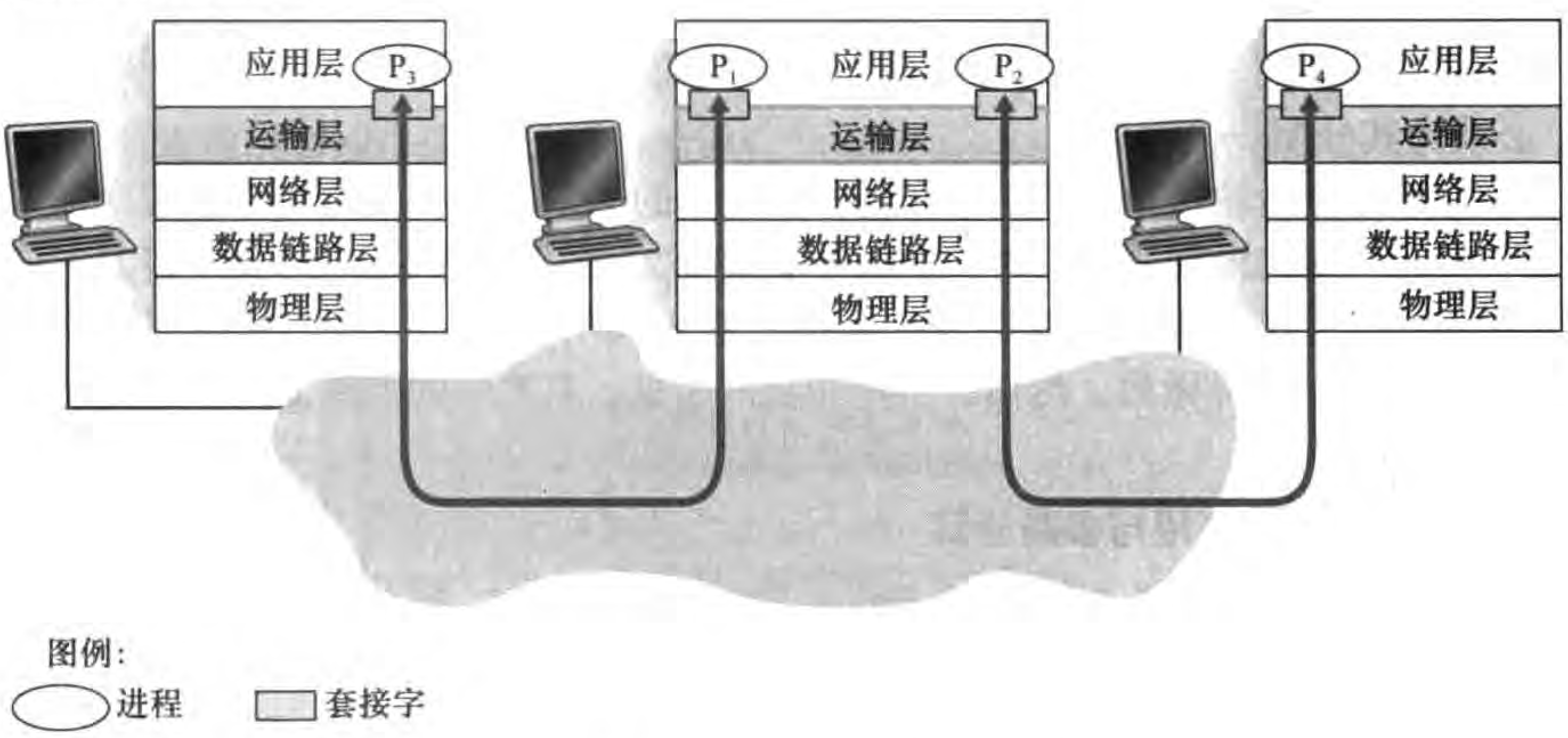


图 3-2 运输层的多路复用与多路分解

现在我们考虑接收主机怎样将一个到达的运输层报文段定向到适当的套接字。为此目的，每个运输层报文段中具有几个字段。在接收端，运输层检查这些字段，标识出接收套接字，进而将报文段定向到该套接字。将运输层报文段中的数据交付到正确的套接字的工作称为**多路分解**（demultiplexing）。在源主机从不同套接字中收集数据块，并为每个数据块封装上首部信息（这将在以后用于分解）从而生成报文段，然后将报文段传递到网络层，所有这些工作称为**多路复用**（multiplexing）。值得注意的是，图 3-2 中的中间那台主机的运输层必须将从其下的网络层收到的报文段分解后交给其上的 P_1 或 P_2 进程；这一过程是通过将到达的报文段数据定向到对应进程的套接字来完成的。中间主机中的运输层也必须收集从这些套接字输出的数据，形成运输层报文段，然后将其向下传递给网络层。尽

管我们在因特网运输层协议的环境下引入了多路复用和多路分解，认识到下列事实是重要的：它们与在某层（在运输层或别处）的单一协议何时被位于接下来的较高层的多个协议使用有关。

为了说明分解的工作过程，再回顾一下前面一节的家庭类比。每一个孩子通过他们的名字来标识。当 Bill 从邮递员处收到一批信件，并通过查看收信人名字而将信件亲手交付给他的兄弟姐妹们时，他执行的就是一个分解操作。当 Ann 从兄弟姐妹们那里收集信件并将它们交给邮递员时，她执行的就是一个多路复用操作。

既然我们理解了运输层多路复用与多路分解的作用，那就再来看看在主机中它们实际是怎样工作的。通过上述讨论，我们知道运输层多路复用要求：①套接字有唯一标识符；②每个报文段有特殊字段来指示该报文段所要交付到的套接字。如图 3-3 所示，这些特殊字段是**源端口号字段**（source port number field）和**目的端口号字段**（destination port number field）。（UDP 报文段和 TCP 报文段还有其他的一些字段，这些将在本章后继几节中进行讨论。）端口号是一个 16 比特的数，其大小在 0～65535 之间。0～1023 范围的端口号称为**周知端口号**（well-known port number），是受限制的，这是指它们保留给诸如 HTTP（它使用端口号 80）和 FTP（它使用端口号 21）之类的周知应用层协议来使用。周知端口的列表在 RFC 1700 中给出，同时在 http://www.iana.org 上有更新文档［RFC 3232］。当我们开发一个新的应用程序时（如在 2.7 节中开发的一个简单应用程序），必须为其分配一个端口号。



图 3-3　运输层报文段中的源与目的端口字段

现在应该清楚运输层是怎样能够实现分解服务的了：在主机上的每个套接字能够分配一个端口号，当报文段到达主机时，运输层检查报文段中的目的端口号，并将其定向到相应的套接字。然后报文段中的数据通过套接字进入其所连接的进程。如我们将看到的那样，UDP 大体上是这样做的。然而，也将如我们所见，TCP 中的多路复用与多路分解更为复杂。

1. 无连接的多路复用与多路分解

2.7.1 节讲过，在主机上运行的 Python 程序使用下面一行代码创建了一个 UDP 套接字：

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

当用这种方式创建一个 UDP 套接字时，运输层自动地为该套接字分配一个端口号。特别是，运输层从范围 1024～65535 内分配一个端口号，该端口号是当前未被该主机中任何其他 UDP 端口使用的号。另外一种方法是，在创建一个套接字后，我们能够在 Python 程序中增加一行代码，通过套接字 bind() 方法为这个 UDP 套接字关联一个特定的端口号（如 19157）：

```
clientSocket.bind(('', 19157))
```

如果应用程序开发者所编写的代码实现的是一个“周知协议”的服务器端，那么开发者就必须为其分配一个相应的周知端口号。通常，应用程序的客户端让运输层自动地（并且是透明地）分配端口号，而服务器端则分配一个特定的端口号。

通过为 UDP 套接字分配端口号，我们现在能够精确地描述 UDP 的复用与分解了。假

定在主机 A 中的一个进程具有 UDP 端口 19157，它要发送一个应用程序数据块给位于主机 B 中的另一进程，该进程具有 UDP 端口 46428。主机 A 中的运输层创建一个运输层报文段，其中包括应用程序数据、源端口号（19157）、目的端口号（46428）和两个其他值（将在后面讨论，它对当前的讨论并不重要）。然后，运输层将得到的报文段传递到网络层。网络层将该报文段封装到一个 IP 数据报中，并尽力而为地将报文段交付给接收主机。如果该报文段到达接收主机 B，接收主机运输层就检查该报文段中的目的端口号（46428）并将该报文段交付给端口号 46428 所标识的套接字。值得注意的是，主机 B 能够运行多个进程，每个进程有自己的 UDP 套接字及相应的端口号。值得注意的是，主机 B 可能运行多个进程，每个进程都具有其自己的 UDP 套接字和相联系的端口号。当 UDP 报文段从网络到达时，主机 B 通过检查该报文段中的目的端口号，将每个报文段定向（分解）到相应的套接字。

注意到下述事实是重要的：一个 UDP 套接字是由一个二元组全面标识的，该二元组包含一个目的 IP 地址和一个目的端口号。因此，如果两个 UDP 报文段有不同的源 IP 地址和/或源端口号，但具有相同的目的 IP 地址和目的端口号，那么这两个报文段将通过相同的目的套接字被定向到相同的目的进程。

你也许现在想知道，源端口号的用途是什么呢？如图 3-4 所示，在 A 到 B 的报文段中，源端口号用作"返回地址"的一部分，即当 B 需要回发一个报文段给 A 时，B 到 A 的报文段中的目的端口号便从 A 到 B 的报文段中的源端口号中取值。（完整的返回地址是 A 的 IP 地址和源端口号。）举一个例子，回想 2.7 节学习过的那个 UDP 服务器程序。在 UDPServer. py 中，服务器使用 recvfrom() 方法从其自客户接收到的报文段中提取出客户端（源）端口号，然后，它将所提取的源端口号作为目的端口号，向客户发送一个新的报文段。

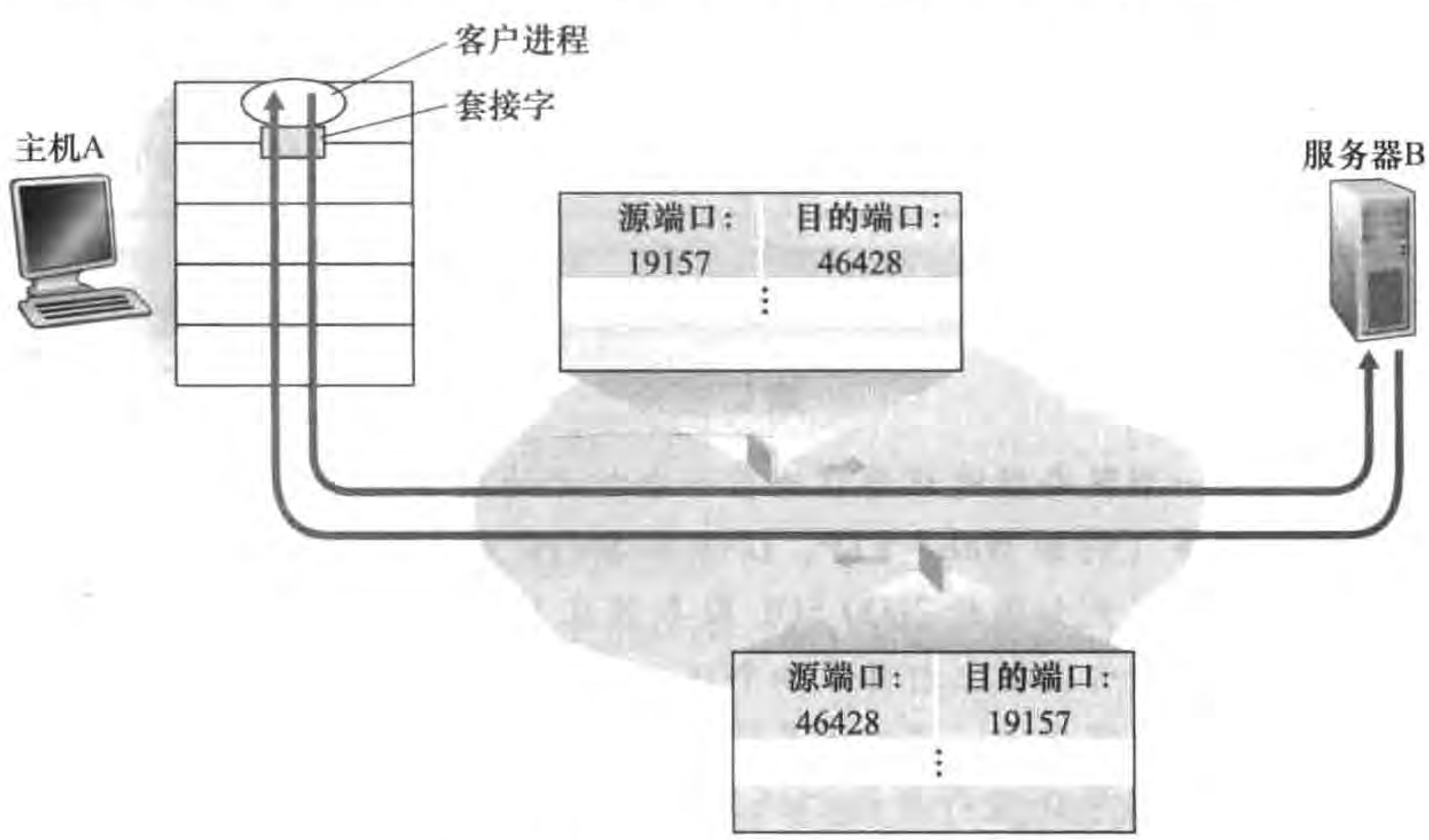


图 3-4 源端口号与目的端口号的反转

2. 面向连接的多路复用与多路分解

为了理解 TCP 多路分解，我们必须更为仔细地研究 TCP 套接字和 TCP 连接创建。TCP 套接字和 UDP 套接字之间的一个细微差别是，TCP 套接字是由一个四元组（源 IP 地址，源端口号，目的 IP 地址，目的端口号）来标识的。因此，当一个 TCP 报文段从网络到达

一台主机时，该主机使用全部4个值来将报文段定向（分解）到相应的套接字。特别与UDP不同的是，两个具有不同源IP地址或源端口号的到达TCP报文段将被定向到两个不同的套接字，除非TCP报文段携带了初始创建连接的请求。为了深入地理解这一点，我们再来重新考虑2.7.2节中的TCP客户-服务器编程的例子：

- TCP服务器应用程序有一个“欢迎套接字”，它在12000号端口上等待来自TCP客户（见图2-27）的连接建立请求。
- TCP客户使用下面的代码创建一个套接字并发送一个连接建立请求报文段：

```
clientSocket = socket(AF_INET, SOCK_STREAM)
               clientSocket.connect((serverName,12000))
```

- 一条连接建立请求只不过是一个目的端口号为12000，TCP首部的特定“连接建立位”置位的TCP报文段（在3.5节进行讨论）。这个报文段也包含一个由客户选择的源端口号。
- 当运行服务器进程的计算机的主机操作系统接收到具有目的端口12000的入连接请求报文段后，它就定位服务器进程，该进程正在端口号12000等待接受连接。该服务器进程则创建一个新的套接字：

```
connectionSocket, addr = serverSocket.accept()
```

- 该服务器的运输层还注意到连接请求报文段中的下列4个值：①该报文段中的源端口号；②源主机IP地址；③该报文段中的目的端口号；④自身的IP地址。新创建的连接套接字通过这4个值来标识。所有后续到达的报文段，如果它们的源端口号、源主机IP地址、目的端口号和目的IP地址都与这4个值匹配，则被分解到这个套接字。随着TCP连接完成，客户和服务器便可相互发送数据了。

服务器主机可以支持很多并行的TCP套接字，每个套接字与一个进程相联系，并由其四元组来标识每个套接字。当一个TCP报文段到达主机时，所有4个字段（源IP地址，源端口，目的IP地址，目的端口）被用来将报文段定向（分解）到相应的套接字。

关注安全性

端口扫描

我们已经看到一个服务器进程潜在地在一个打开的端口等待远程客户的接触。某些端口为周知应用程序（例如Web、FTP、DNS和SMTP服务器）所预留；依照惯例其他端口由流行应用程序（例如微软2000 SQL服务器在UDP 1434端口上监听请求）使用。因此，如果我们确定一台主机上打开了一个端口，也许就能够将该端口映射到在该主机运行的一个特定的应用程序上。这对于系统管理员非常有用，系统管理员通常希望知晓有什么样的网络应用程序正运行在他们的网络主机上。而攻击者为了“寻找突破口”，也要知道在目标主机上有哪些端口打开。如果发现一台主机正在运行具有已知安全缺陷的应用程序（例如，在端口1434上监听的一台SQL服务器会遭受缓存溢出，使得一个远程用户能在易受攻击的主机上执行任意代码，这是一种由Slammer蠕虫所利用的缺陷[CERT 2003-04]），那么该主机已成为攻击者的囊中之物了。

确定哪个应用程序正在监听哪些端口是一件相对容易的事情。事实上有许多公共域程序（称为端口扫描器）做的正是这种事情。也许它们之中使用最广泛的是nmap，该程

序在 http://nmap.org/上免费可用，并且包括在大多数 Linux 分发软件中。对于 TCP，nmap 顺序地扫描端口，寻找能够接受 TCP 连接的端口。对于 UDP，nmap 也是顺序地扫描端口，寻找对传输的 UDP 报文段进行响应的 UDP 端口。在这两种情况下，nmap 返回打开的、关闭的或不可达的端口列表。运行 nmap 的主机能够尝试扫描因特网中**任何地方**的目的主机。我们将在 3.5.6 节中再次用到 nmap，在该节中我们将讨论 TCP 连接管理。

图 3-5 图示了这种情况，图中主机 C 向服务器 B 发起了两个 HTTP 会话，主机 A 向服务器 B 发起了一个 HTTP 会话。主机 A 与主机 C 及服务器 B 都有自己唯一的 IP 地址，它们分别是 A、C、B。主机 C 为其两个 HTTP 连接分配了两个不同的源端口号（26145 和 7532）。因为主机 A 选择源端口号时与主机 C 互不相干，因此它也可以将源端口号 26145 分配给其 HTTP 连接。但这不是问题，即服务器 B 仍然能够正确地分解这两个具有相同源端口号的连接，因为这两条连接有不同的源 IP 地址。

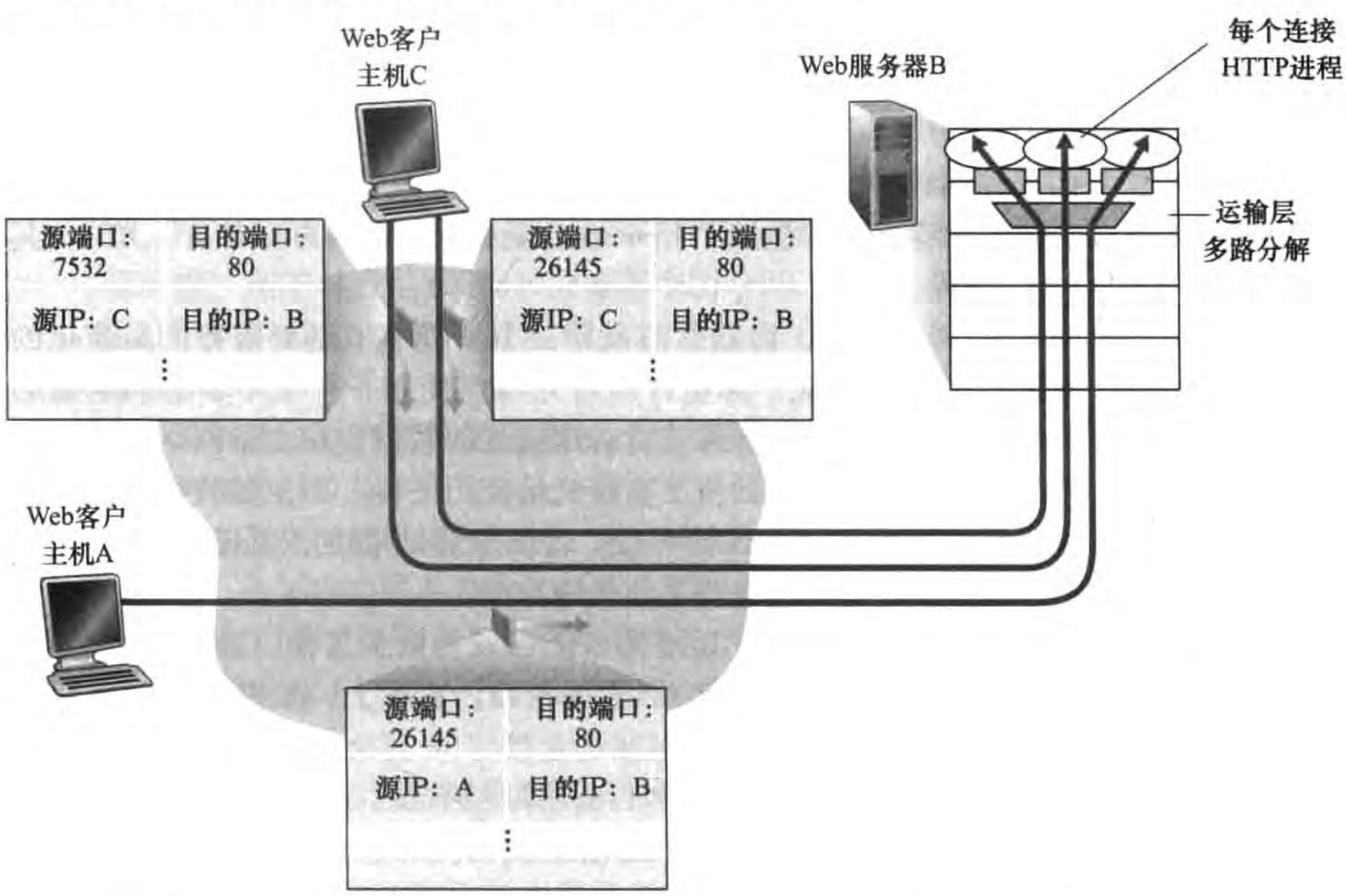


图 3-5 两个客户使用相同的目的端口号（80）与同一个 Web 服务器应用通信

3. Web 服务器与 TCP

在结束这个讨论之前，再多说几句 Web 服务器以及它们如何使用端口号是有益的。考虑一台运行 Web 服务器的主机，例如在端口 80 上运行一个 Apache Web 服务器。当客户（如浏览器）向该服务器发送报文段时，所有报文段的目的端口都将为 80。特别是，初始连接建立报文段和承载 HTTP 请求的报文段都有 80 的目的端口。如我们刚才描述的那样，该服务器能够根据源 IP 地址和源端口号来区分来自不同客户的报文段。

图 3-5 显示了一台 Web 服务器为每条连接生成一个新进程。如图 3-5 所示，每个这样的进程都有自己的连接套接字，通过这些套接字可以收到 HTTP 请求和发送 HTTP 响应。

然而，我们要提及的是，连接套接字与进程之间并非总是有着一一对应的关系。事实上，当今的高性能 Web 服务器通常只使用一个进程，但是为每个新的客户连接创建一个具有新连接套接字的新线程。（线程可被看作是一个轻量级的子进程。）如果做了第2章的第一个编程作业，你所构建的 Web 服务器就是这样工作的。对于这样一台服务器，在任意给定的时间内都可能有（具有不同标识的）许多连接套接字连接到相同的进程。

如果客户与服务器使用持续 HTTP，则在整条连接持续期间，客户与服务器之间经由同一个服务器套接字交换 HTTP 报文。然而，如果客户与服务器使用非持续 HTTP，则对每一对请求/响应都创建一个新的 TCP 连接并在随后关闭，因此对每一对请求/响应创建一个新的套接字并在随后关闭。这种套接字的频繁创建和关闭会严重地影响一个繁忙的 Web 服务器的性能（尽管有许多操作系统技巧可用来减轻这个问题的影响）。读者若对与持续和非持续 HTTP 有关的操作系统问题感兴趣的话，可参见 [Nielsen 1997，Nahum 2002]。

既然我们已经讨论过了运输层多路复用与多路分解问题，下面我们就继续讨论因特网运输层协议之一，即 UDP。在下一节中，我们将看到 UDP 无非就是对网络层协议增加了一点（多路）复用/（多路）分解服务而已。

3.3 无连接运输：UDP

在本节中，我们要仔细地研究一下 UDP，看它是怎样工作的，能做些什么。我们鼓励你回过来看一下2.1节的内容，其中包括了 UDP 服务模型的概述，再看看2.7.1节，其中讨论了 UDP 上的套接字编程。

为了激发我们讨论 UDP 的热情，假如你对设计一个不提供不必要服务的最简化的运输层协议感兴趣。你将打算怎样做呢？你也许会首先考虑使用一个无所事事的运输层协议。特别是在发送方一侧，你可能会考虑将来自应用进程的数据直接交给网络层；在接收方一侧，你可能会考虑将从网络层到达的报文直接交给应用进程。而正如我们在前一节所学的，我们必须做一点点事，而不是什么都不做！运输层最低限度必须提供一种复用/分解服务，以便在网络层与正确的应用级进程之间传递数据。

由 [RFC 768] 定义的 UDP 只是做了运输协议能够做的最少工作。除了复用/分解功能及少量的差错检测外，它几乎没有对 IP 增加别的东西。实际上，如果应用程序开发人员选择 UDP 而不是 TCP，则该应用程序差不多就是直接与 IP 打交道。UDP 从应用进程得到数据，附加上用于多路复用/分解服务的源和目的端口号字段，以及两个其他的小字段，然后将形成的报文段交给网络层。网络层将该运输层报文段封装到一个 IP 数据报中，然后尽力而为地尝试将此报文段交付给接收主机。如果该报文段到达接收主机，UDP 使用目的端口号将报文段中的数据交付给正确的应用进程。值得注意的是，使用 UDP 时，在发送报文段之前，发送方和接收方的运输层实体之间没有握手。正因为如此，UDP 被称为是*无连接的*。

DNS 是一个通常使用 UDP 的应用层协议的例子。当一台主机中的 DNS 应用程序想要进行一次查询时，它构造了一个 DNS 查询报文并将其交给 UDP。无须执行任何与运行在目的端系统中的 UDP 实体之间的握手，主机端的 UDP 为此报文添加首部字段，然后将形成的报文段交给网络层。网络层将此 UDP 报文段封装进一个 IP 数据报中，然后将其发送给一个名字服务器。在查询主机中的 DNS 应用程序则等待对该查询的响应。如果它没有收到响应（可能是由于底层网络丢失了查询或响应），则要么试图向另一个名字服务器发送该查询，要么通知调用的应用程序它不能获得响应。

现在你也许想知道，为什么应用开发人员宁愿在UDP之上构建应用，而不选择在TCP上构建应用？既然TCP提供了可靠数据传输服务，而UDP不能提供，那么TCP是否总是首选的呢？答案是否定的，因为有许多应用更适合用UDP，原因主要以下几点：

- *关于发送什么数据以及何时发送的应用层控制更为精细*。采用UDP时，只要应用进程将数据传递给UDP，UDP就会将此数据打包进UDP报文段并立即将其传递给网络层。在另一方面，TCP有一个拥塞控制机制，以便当源和目的主机间的一条或多条链路变得极度拥塞时来遏制运输层TCP发送方。TCP仍将继续重新发送数据报文段直到目的主机收到此报文并加以确认，而不管可靠交付需要用多长时间。因为实时应用通常要求最小的发送速率，不希望过分地延迟报文段的传送，且能容忍一些数据丢失，TCP服务模型并不是特别适合这些应用的需要。如后面所讨论的，这些应用可以使用UDP，并作为应用的一部分来实现所需的、超出UDP的不提供不必要的报文段交付服务之外的额外功能。
- *无须连接建立*。如我们后面所讨论的，TCP在开始数据传输之前要经过三次握手。UDP却不需要任何准备即可进行数据传输。因此UDP不会引入建立连接的时延。这可能是DNS运行在UDP之上而不是运行在TCP之上的主要原因（如果运行在TCP上，则DNS会慢得多）。HTTP使用TCP而不是UDP，因为对于具有文本数据的Web网页来说，可靠性是至关重要的。但是，如我们在2.2节中简要讨论的那样，HTTP中的TCP连接建立时延对于与下载Web文档相关的时延来说是一个重要因素。用于谷歌的Chrome浏览器中的QUIC协议（快速UDP因特网连接［Iyengar 2015］）将UDP作为其支撑运输协议并在UDP之上的应用层协议中实现可靠性。
- *无连接状态*。TCP需要在端系统中维护连接状态。此连接状态包括接收和发送缓存、拥塞控制参数以及序号与确认号的参数。我们将在3.5节看到，要实现TCP的可靠数据传输服务并提供拥塞控制，这些状态信息是必要的。另一方面，UDP不维护连接状态，也不跟踪这些参数。因此，某些专门用于某种特定应用的服务器当应用程序运行在UDP之上而不是运行在TCP上时，一般都能支持更多的活跃客户。
- *分组首部开销小*。每个TCP报文段都有20字节的首部开销，而UDP仅有8字节的开销。

图3-6列出了流行的因特网应用及其所使用的运输协议。如我们所期望的那样，电子邮件、远程终端访问、Web及文件传输都运行在TCP之上。因为所有这些应用都需要TCP的可靠数据传输服务。无论如何，有很多重要的应用是运行在UDP上而不是TCP上。例如，UDP用于承载网络管理数据（SNMP，参见5.7节）。在这种场合下，UDP要优于TCP，因为网络管理应用程序通常必须在该网络处于重压状态时运行，而正是在这个时候可靠的、拥塞受控的数据传输难以实现。此外，如我们前面所述，DNS运行在UDP之上，从而避免了TCP的连接创建时延。

如图3-6所示，UDP和TCP现在都用于多媒体应用，如因特网电话、实时视频会议、流式存储音频与视频。我们将在第9章仔细学习这些应用。我们刚说过，既然所有这些应用都能容忍少量的分组丢失，因此可靠数据传输对于这些应用的成功并不是至关重要的。此外，TCP的拥塞控制会导致如因特网电话、视频会议之类的实时应用性能变得很差。由于这些原因，多媒体应用开发人员通常将这些应用运行在UDP之上而不是TCP之上。当分组丢包率低时，并且为了安全原因，某些机构阻塞UDP流量（参见第8章），对于流式

媒体传输来说，TCP变得越来越有吸引力了。

应用	应用层协议	下面的运输协议
电子邮件	SMTP	TCP
远程终端访问	Telnet	TCP
Web	HTTP	TCP
文件传输	FTP	TCP
远程文件服务器	NFS	通常UDP
流式多媒体	通常专用	UDP或TCP
因特网电话	通常专用	UDP或TCP
网络管理	SNMP	通常UDP
名字转换	DNS	通常UDP

图3-6 流行的因特网应用及其下面的运输协议

虽然目前通常这样做，但在UDP之上运行多媒体应用是有争议的。如我们前面所述，UDP没有拥塞控制。但是，需要拥塞控制来预防网络进入一种拥塞状态，在拥塞状态中可做的有用工作非常少。如果每个人都启动流式高比特率视频而不使用任何拥塞控制的话，就会使路由器中有大量的分组溢出，以至于非常少的UDP分组能成功地通过源到目的的路径传输。况且，由无控制的UDP发送方引入的高丢包率将引起TCP发送方（如我们将看到的那样，TCP遇到拥塞将减小它们的发送速率）大大地减小它们的速率。因此，UDP中缺乏拥塞控制能够导致UDP发送方和接收方之间的高丢包率，并挤垮了TCP会话，这是一个潜在的严重问题［Floyd 1999］。很多研究人员已提出了一些新机制，以促使所有的数据源（包括UDP源）执行自适应的拥塞控制［Mahdavi 1997；Floyd 2000；Kohler 2006；RFC 4340］。

在讨论UDP报文段结构之前，我们要提一下，使用UDP的应用是可能实现可靠数据传输的。这可通过在应用程序自身中建立可靠性机制来完成（例如，可通过增加确认与重传机制来实现，如采用我们将在下一节学习的一些机制）。我们前面讲过在谷歌的Chrome浏览器中所使用的QUIC协议［Iyengar 2015］在UDP之上的应用层协议中实现了可靠性。但这并不是无足轻重的任务，它会使应用开发人员长时间地忙于调试。无论如何，将可靠性直接构建于应用程序中可以使其“左右逢源”。也就是说应用进程可以进行可靠通信，而无须受制于由TCP拥塞控制机制强加的传输速率限制。

3.3.1 UDP报文段结构



图3-7 UDP报文段结构

UDP报文段结构如图3-7所示，它由RFC 768定义。应用层数据占用UDP报文段的数据字段。例如，对于DNS应用，数据字段要么包含一个查询报文，要么包含一个响应报文。对于流式音频应用，音频抽样数据填充到数据字段。UDP首部只有4个字段，每个字段由两个字节组成。如前一节所讨论的，通过端口号可以使目的主机将应用数据交给运行在目的端系统中的相应进程（即执行分解功能）。长度字段指示了在UDP报文段中的字节数（首部加数据）。因为数据字段的长度在一个UDP段中不同于在另一个段中，故需要一个明确的长度。接收方使用检验和来检查在该报文段中是

否出现了差错。实际上，计算检验和时，除了 UDP 报文段以外还包括了 IP 首部的一些字段。但是我们忽略这些细节，以便能从整体上看问题。下面我们将讨论检验和的计算。在 6.2 节中将描述差错检测的基本原理。长度字段指明了包括首部在内的 UDP 报文段长度（以字节为单位）。

3.3.2 UDP 检验和

UDP 检验和提供了差错检测功能。这就是说，检验和用于确定当 UDP 报文段从源到达目的地移动时，其中的比特是否发生了改变（例如，由于链路中的噪声干扰或者存储在路由器中时引入问题）。发送方的 UDP 对报文段中的所有 16 比特字的和进行反码运算，求和时遇到的任何溢出都被回卷。得到的结果被放在 UDP 报文段中的检验和字段。下面给出一个计算检验和的简单例子。在 RFC 1071 中可以找到有效实现的细节，还可在［Stone 1998；Stone 2000］中找到它处理真实数据的性能。举例来说，假定我们有下面 3 个 16 比特的字：

```
0110011001100000
0101010101010101
1000111100001100
```

这些 16 比特字的前两个之和是：

```
 0110011001100000
 0101010101010101
 ----------------
 1011101110110101
```

再将上面的和与第三个字相加，得出：

```
 1011101110110101
 1000111100001100
 ----------------
 0100101011000010
```

注意到最后一次加法有溢出，它要被回卷。反码运算就是将所有的 0 换成 1，所有的 1 转换成 0。因此，该和 0100101011000010 的反码运算结果是 1011010100111101，这就变为了检验和。在接收方，全部的 4 个 16 比特字（包括检验和）加在一起。如果该分组中没有引入差错，则显然在接收方处该和将是 1111111111111111。如果这些比特之一是 0，那么我们就知道该分组中已经出现了差错。

你可能想知道为什么 UDP 首先提供了检验和，就像许多链路层协议（包括流行的以太网协议）也提供了差错检测那样。其原因是不能保证源和目的之间的所有链路都提供差错检测；这就是说，也许这些链路中的一条可能使用没有差错检测的协议。此外，即使报文段经链路正确地传输，当报文段存储在某台路由器的内存中时，也可能引入比特差错。在既无法确保逐链路的可靠性，又无法确保内存中的差错检测的情况下，如果端到端数据传输服务要提供差错检测，UDP 就必须在*端到端*基础上在运输层提供差错检测。这是一个在系统设计中被称颂的**端到端原则**（end-end principle）的例子［Saltzer 1984］，该原则表述为因为某种功能（在此时为差错检测）必须基于端到端实现："与在较高级别提供这些功能的代价相比，在较低级别上设置的功能可能是冗余的或几乎没有价值的。"

因为假定 IP 是可以运行在任何第二层协议之上的，运输层提供差错检测作为一种保险措施是非常有用的。虽然 UDP 提供差错检测，但它对差错恢复无能为力。UDP 的某种

实现只是丢弃受损的报文段；其他实现是将受损的报文段交给应用程序并给出警告。

至此结束了关于 UDP 的讨论。我们将很快看到 TCP 为应用提供了可靠数据传输及 UDP 所不能提供的其他服务。TCP 自然要比 UDP 复杂得多。然而，在讨论 TCP 之前，我们后退一步，先来讨论一下可靠数据传输的基本原理是有用的。

3.4 可靠数据传输原理

在本节中，我们在一般场景下考虑可靠数据传输的问题。因为可靠数据传输的实现问题不仅在运输层出现，也会在链路层以及应用层出现，这时讨论它是恰当的。因此，一般性问题对网络来说更为重要。如果的确要将所有网络中最为重要的“前 10 个”问题排名的话，可靠数据传输将是名列榜首的候选者。在下一节中，我们将学习 TCP，尤其要说明 TCP 所采用的许多原理，而这些正是我们打算描述的内容。

图 3-8 图示说明了我们学习可靠数据传输的框架。为上层实体提供的服务抽象是：数据可以通过一条可靠的信道进行传输。借助于可靠信道，传输数据比特就不会受到损坏（由 0 变为 1，或者相反）或丢失，而且所有数据都是按照其发送顺序进行交付。这恰好就是 TCP 向调用它的因特网应用所提供的服务模型。

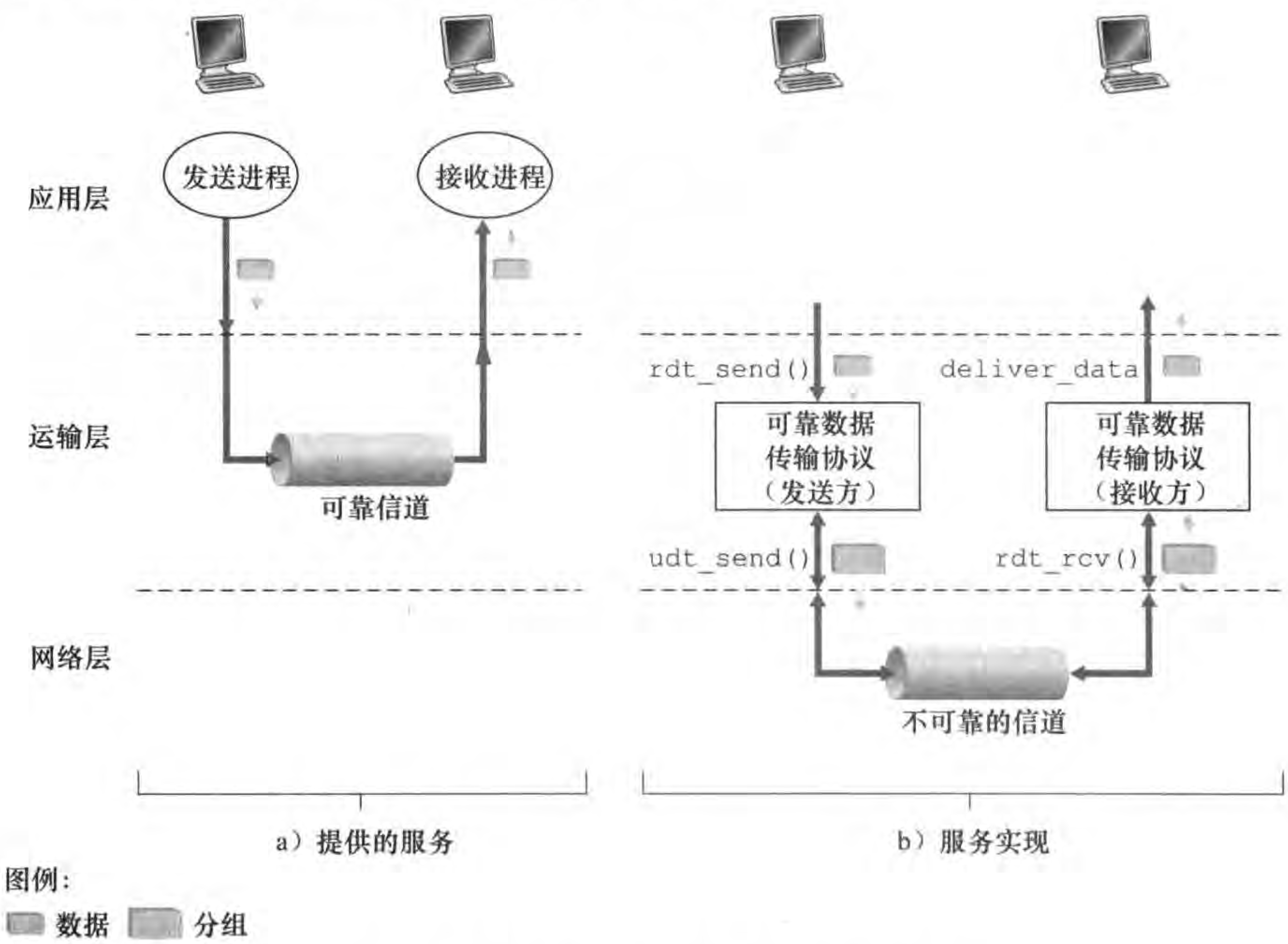


图 3-8 可靠数据传输：服务模型与服务实现

实现这种服务抽象是**可靠数据传输协议**（reliable data transfer protocol）的责任。由于可靠数据传输协议的下层协议也许是不可靠的，因此这是一项困难的任务。例如，TCP 是在不可靠的（IP）端到端网络层之上实现的可靠数据传输协议。更一般的情况是，两个可靠通信端点的下层可能是由一条物理链路（如在链路级数据传输协议的场合下）组成或是由一个全球互联网络（如在运输级协议的场合下）组成。然而，就我们的目的而言，我们

可将较低层直接视为不可靠的点对点信道。

在本节中，考虑到底层信道模型越来越复杂，我们将不断地开发一个可靠数据传输协议的发送方一侧和接收方一侧。例如，我们将考虑当底层信道能够损坏比特或丢失整个分组时，需要什么样的协议机制。这里贯穿我们讨论始终的一个假设是分组将以它们发送的次序进行交付，某些分组可能会丢失；这就是说，底层信道将不会对分组重排序。图3-8b图示说明了用于数据传输协议的接口。通过调用 rdt_send() 函数，上层可以调用数据传输协议的发送方。它将要发送的数据交付给位于接收方的较高层。（这里 rdt 表示可靠数据传输协议，_send 指示 rdt 的发送端正在被调用。开发任何协议的第一步就是要选择一个好的名字！）在接收端，当分组从信道的接收端到达时，将调用 rdt_rcv()。当 rdt 协议想要向较高层交付数据时，将通过调用 deliver_data() 来完成。后面，我们将使用术语“分组”而不用运输层的“报文段”。因为本节研讨的理论适用于一般的计算机网络，而不只是用于因特网运输层，所以这时采用通用术语“分组”也许更为合适。

在本节中，我们仅考虑**单向数据传输**（unidirectional data transfer）的情况，即数据传输是从发送端到接收端的。可靠的**双向数据传输**（bidirectional data transfer）（即全双工数据传输）情况从概念上讲不会更难，但解释起来更为单调乏味。虽然我们只考虑单向数据传输，注意到下列事实是重要的，我们的协议也需要在发送端和接收端两个方向上传输分组，如图3-8所示。我们很快会看到，除了交换含有待传送的数据的分组之外，rdt 的发送端和接收端还需往返交换控制分组。rdt 的发送端和接收端都要通过调用 udt_send() 发送分组给对方（其中 udt 表示不可靠数据传输）。

3.4.1 构造可靠数据传输协议

我们现在一步步地研究一系列协议，它们一个比一个更为复杂，最后得到一个完美、可靠的数据传输协议。

1. 经完全可靠信道的可靠数据传输：rdt1.0

首先，我们考虑最简单的情况，即底层信道是完全可靠的。我们称该协议为 rdt1.0，该协议本身是简单的。图3-9显示了 rdt1.0 发送方和接收方的**有限状态机**（Finite-State Machine，FSM）的定义。图3-9a 中的 FSM 定义了发送方的操作，图3-9b 中的 FSM 定义了接收方的操作。注意到下列问题是重要的，发送方和接收方有各自的 FSM。图3-9中发送方和接收方的 FSM 每个都只有一个状态。FSM 描述图中的箭头指示了协议从一个状态变迁到另一个状态。（因为图3-9中的每个 FSM 都只有一个状态，因此变迁必定是从一个状态返回到自身；我们很快将看到更复杂的状态图。）引起变迁的事件显示在表示变迁的横线上方，事件发生时所采取的动作显示在横线下方。如果对一个事件没有动作，或没有就事件发生而采取了一个动作，我们将在横线上方或下方使用符号 Λ，以分别明确地表示缺少动作或事件。FSM 的初始状态用



a）rdt1.0：发送端



b）rdt1.0：接收端

图3-9 rdt1.0：用于完全可靠信道的协议

虚线表示。尽管图 3-9 中的 FSM 只有一个状态，但马上我们就将看到多状态的 FSM，因此标识每个 FSM 的初始状态是非常重要的。

rdt 的发送端只通过 rdt_send(data)事件接受来自较高层的数据，产生一个包含该数据的分组（经由 make_pkt(data)动作），并将分组发送到信道中。实际上，rdt_send(data)事件是由较高层应用的过程调用产生的（例如，rdt_send()）。

在接收端，rdt 通过 rdt_rcv(packet）事件从底层信道接收一个分组，从分组中取出数据（经由 extract(packet，data）动作），并将数据上传给较高层（通过 deliver_data(data)动作）。实际上，rdt_rcv(packet）事件是由较低层协议的过程调用产生的（例如，rdt_rcv()）。

在这个简单的协议中，一个单元数据与一个分组没差别。而且，所有分组是从发送方流向接收方；有了完全可靠的信道，接收端就不需要提供任何反馈信息给发送方，因为不必担心出现差错！注意到我们也已经假定了接收方接收数据的速率能够与发送方发送数据的速率一样快。因此，接收方没有必要请求发送方慢一点！

2. 经具有比特差错信道的可靠数据传输：rdt2.0

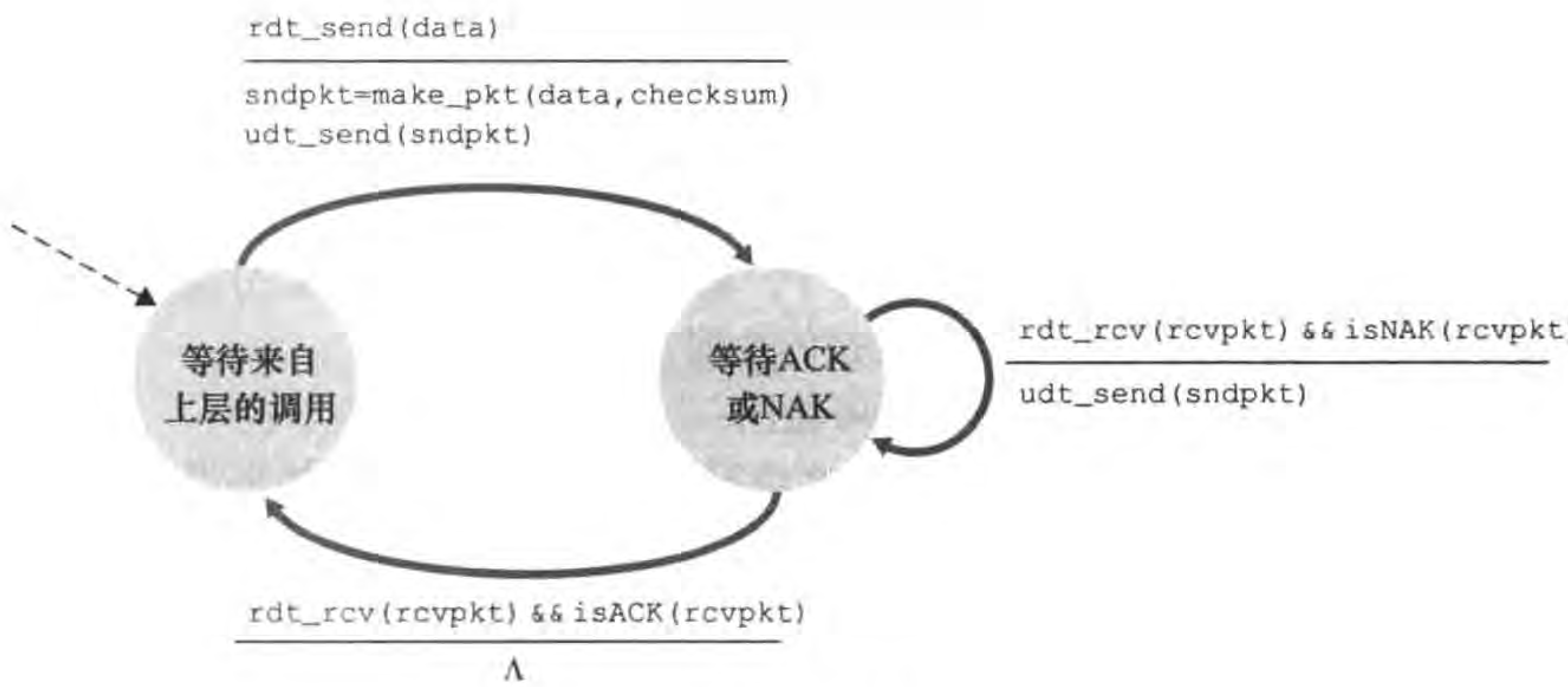
底层信道更为实际的模型是分组中的比特可能受损的模型。在分组的传输、传播或缓存的过程中，这种比特差错通常会出现在网络的物理部件中。我们眼下还将继续假定所有发送的分组（虽然有些比特可能受损）将按其发送的顺序被接收。

在研发一种经这种信道进行可靠通信的协议之前，首先考虑一下人们会怎样处理这类情形。考虑一下你自己是怎样通过电话口述一条长报文的。在通常情况下，报文接收者在听到、理解并记下每句话后可能会说“OK”。如果报文接收者听到一句含糊不清的话时，他可能要求你重复那句容易误解的话。这种口述报文协议使用了**肯定确认**（positive acknowledgment）（“OK”）与**否定确认**（negative acknowledgment）（“请重复一遍”）。这些控制报文使得接收方可以让发送方知道哪些内容被正确接收，哪些内容接收有误并因此需要重复。在计算机网络环境中，基于这样重传机制的可靠数据传输协议称为**自动重传请求**（Automatic Repeat reQuest，ARQ）**协议**。

重要的是，ARQ 协议中还需要另外三种协议功能来处理存在比特差错的情况：

- *差错检测*。首先，需要一种机制以使接收方检测到何时出现了比特差错。前一节讲到，UDP 使用因特网检验和字段正是为了这个目的。在第 5 章中，我们将更详细地学习差错检测和纠错技术。这些技术使接收方可以检测并可能纠正分组中的比特差错。此刻，我们只需知道这些技术要求有额外的比特（除了待发送的初始数据比特之外的比特）从发送方发送到接收方；这些比特将被汇集在 rdt2.0 数据分组的分组检验和字段中。
- *接收方反馈*。因为发送方和接收方通常在不同端系统上执行，可能相隔数千英里，发送方要了解接收方情况（此时为分组是否被正确接收）的唯一途径就是让接收方提供明确的反馈信息给发送方。在口述报文情况下回答的“肯定确认”（ACK）和“否定确认”（NAK）就是这种反馈的例子。类似地，我们的 rdt2.0 协议将从接收方向发送方回送 ACK 与 NAK 分组。理论上，这些分组只需要一个比特长；如用 0 表示 NAK，用 1 表示 ACK。
- *重传*。接收方收到有差错的分组时，发送方将重传该分组文。

图 3-10 说明了表示 rdt2.0 的 FSM，该数据传输协议采用了差错检测、肯定确认与否定确认。



a) rdt2.0: 发送端



b) rdt2.0: 接收端

图 3-10 rdt2.0：用于具有比特差错信道的协议

rdt2.0 的发送端有两个状态。在最左边的状态中，发送端协议正等待来自上层传下来的数据。当 rdt_send(data)事件出现时，发送方将产生一个包含待发送数据的分组（sndpkt），带有检验和（例如，就像在 3.3.2 节讨论的对 UDP 报文段使用的方法），然后经由 udt_send(sndpkt)操作发送该分组。在最右边的状态中，发送方协议等待来自接收方的 ACK 或 NAK 分组。如果收到一个 ACK 分组（图 3-10 中符号 rdt_rcv(rcvpkt)&& isACK(rcvpkt)对应该事件），则发送方知道最近发送的分组已被正确接收，因此协议返回到等待来自上层的数据的状态。如果收到一个 NAK 分组，该协议重传上一个分组并等待接收方为响应重传分组而回送的 ACK 和 NAK。注意到下列事实很重要：当发送方处于等待 ACK 或 NAK 的状态时，它不能从上层获得更多的数据；这就是说，rdt_send()事件不可能出现；仅当接收到 ACK 并离开该状态时才能发生这样的事件。因此，发送方将不会发送一块新数据，除非发送方确信接收方已正确接收当前分组。由于这种行为，rdt2.0 这样的协议被称为**停等**（stop-and-wait）协议。

rdt2.0 接收方的 FSM 仍然只有单一状态。当分组到达时，接收方要么回答一个 ACK，要么回答一个 NAK，这取决于收到的分组是否受损。在图 3-10 中，符号 rdt_rcv(rcvpkt)&& corrupt(rcvpkt)对应于收到一个分组并发现有错的事件。

rdt2.0 协议看起来似乎可以运行了，但遗憾的是，它存在一个致命的缺陷。尤其是我们没有考虑到 ACK 或 NAK 分组受损的可能性！（在继续研究之前，你应该考虑怎样解决该问题。）遗憾的是，我们细小的疏忽并非像它看起来那么无关紧要。至少，我们需要在 ACK/NAK 分组中添加检验和比特以检测这样的差错。更难的问题是协议应该怎样纠正 ACK 或 NAK 分组中的差错。这里的难点在于，如果一个 ACK 或 NAK 分组受损，发送方无法知道接收方是否正确接收了上一块发送的数据。

考虑处理受损 ACK 和 NAK 时的 3 种可能性：

- 对于第一种可能性，考虑在口述报文情况下人可能的做法。如果说话者不理解来自接收方回答的“OK”或“请重复一遍”，说话者将可能问“你说什么?”（因此在我们的协议中引入了一种新型发送方到接收方的分组）。接收方则将复述其回答。但是如果说话者的“你说什么?”产生了差错，情况又会怎样呢？接收者不明白那句混淆的话是口述内容的一部分还是一个要求重复上次回答的请求，很可能回一句“你说什么?”。于是，该回答可能含糊不清了。显然，我们走上了一条困难重重之路。
- 第二种可能性是增加足够的检验和比特，使发送方不仅可以检测差错，还可恢复差错。对于会产生差错但不丢失分组的信道，这就可以直接解决问题。
- 第三种方法是，当发送方收到含糊不清的 ACK 或 NAK 分组时，只需重传当前数据分组即可。然而，这种方法在发送方到接收方的信道中引入了**冗余分组**（duplicate packet）。冗余分组的根本困难在于接收方不知道它上次所发送的 ACK 或 NAK 是否被发送方正确地收到。因此它无法事先知道接收到的分组是新的还是一次重传！

解决这个新问题的一个简单方法（几乎所有现有的数据传输协议中，包括 TCP，都采用了这种方法）是在数据分组中添加一新字段，让发送方对其数据分组编号，即将发送数据分组的**序号**（sequence number）放在该字段。于是，接收方只需要检查序号即可确定收到的分组是否一次重传。对于停等协议这种简单情况，1 比特序号就足够了，因为它可让接收方知道发送方是否正在重传前一个发送分组（接收到的分组序号与最近收到的分组序号相同），或是一个新分组（序号变化了，用模 2 运算“前向”移动）。因为目前我们假定信道不丢分组，ACK 和 NAK 分组本身不需要指明它们要确认的分组序号。发送方知道所接收到的 ACK 和 NAK 分组（无论是否是含糊不清的）是为响应其最近发送的数据分组而生成的。

图 3-11 和图 3-12 给出了对 rdt2.1 的 FSM 描述，这是 rdt2.0 的修订版。rdt2.1 的发送方和接收方 FSM 的状态数都是以前的两倍。这是因为协议状态此时必须反映出目前（由发送方）正发送的分组或（在接收方）希望接收的分组的序号是 0 还是 1。值得注意的是，发送或期望接收 0 号分组的状态中的动作与发送或期望接收 1 号分组的状态中的动作是相似的；唯一的不同是序号处理的方法不同。

协议 rdt2.1 使用了从接收方到发送方的肯定确认和否定确认。当接收到失序的分组时，接收方对所接收的分组发送一个肯定确认。如果收到受损的分组，则接收方将发送一个否定确认。如果不发送 NAK，而是对上次正确接收的分组发送一个 ACK，我们也能实现与 NAK 一样的效果。发送方接收到对同一个分组的两个 ACK（即接收**冗余 ACK**（duplicate ACK））后，就知道接收方没有正确接收到跟在被确认两次的分组后面的分组。rdt2.2 是在有比特差错信道上实现的一个无 NAK 的可靠数据传输协议，如图 3-13 和

图3-14所示。rdt2.1 和 rdt2.2 之间的细微变化在于，接收方此时必须包括由一个 ACK 报文所确认的分组序号（这可以通过在接收方 FSM 中，在 make_pkt()中包括参数 ACK 0 或 ACK 1 来实现），发送方此时必须检查接收到的 ACK 报文中被确认的分组序号（这可通过在发送方 FSM 中，在 isACK()中包括参数0或1来实现）。

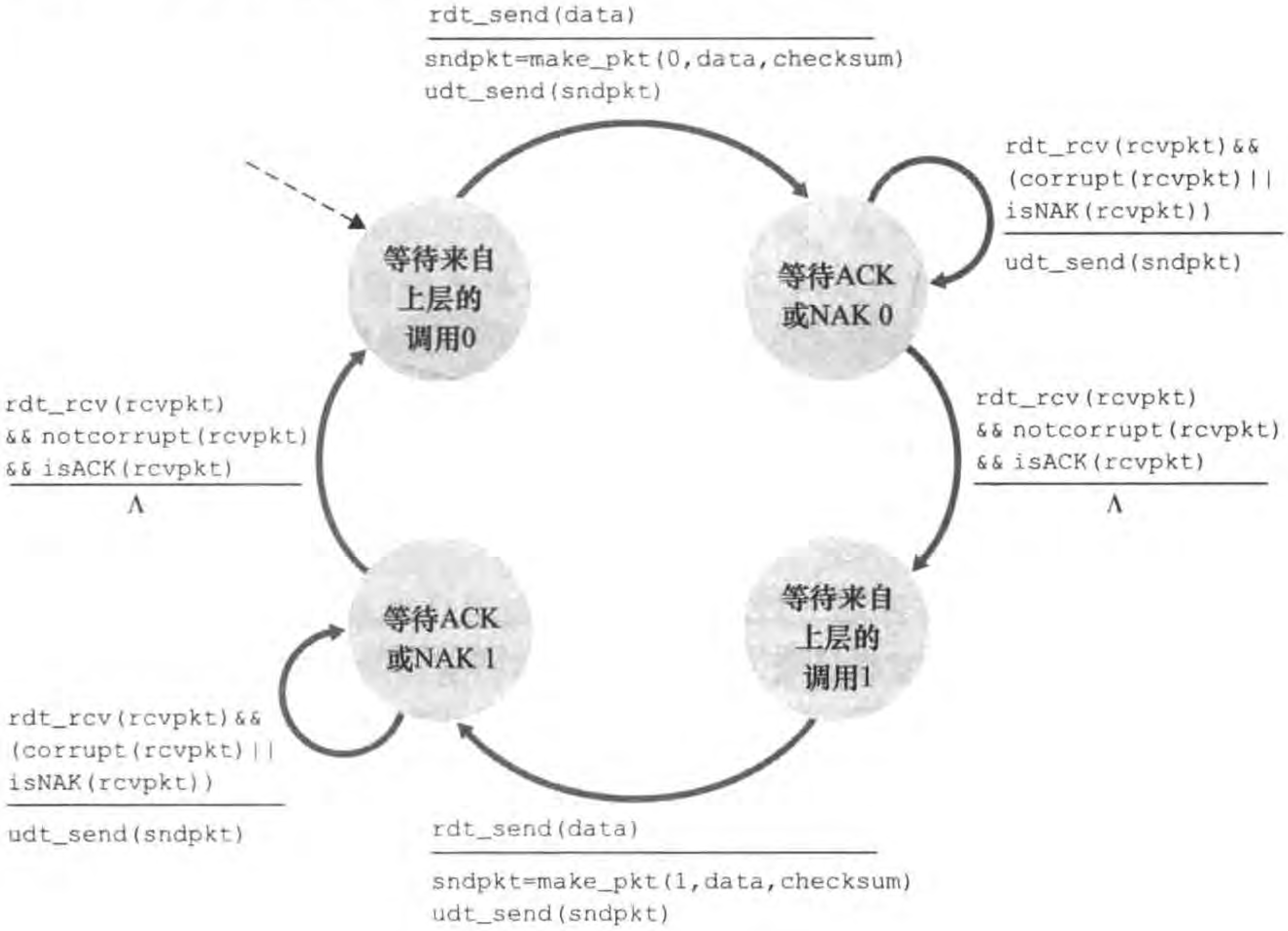


图3-11 rdt2.1 发送方

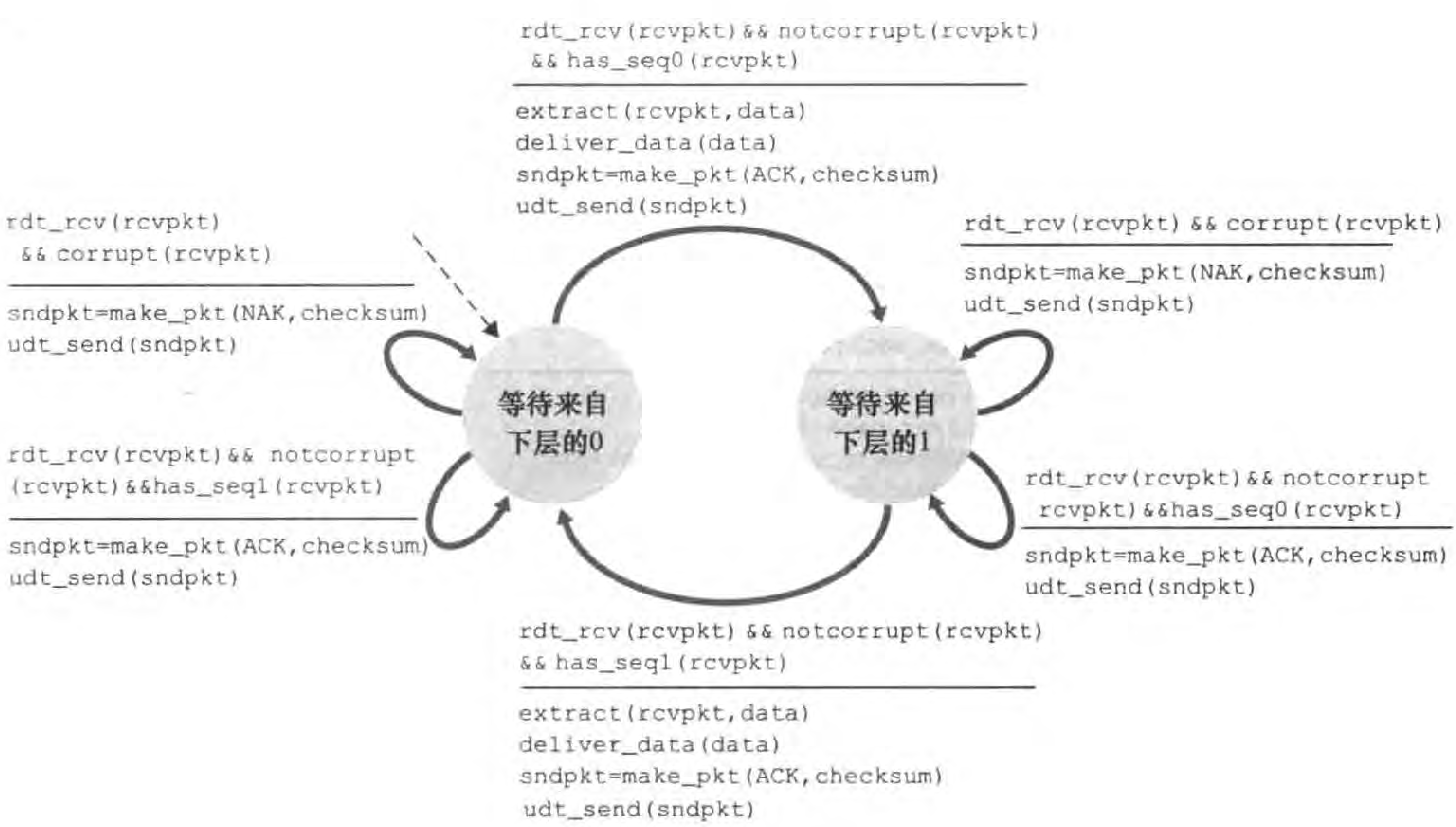


图3-12 rdt2.1 接收方

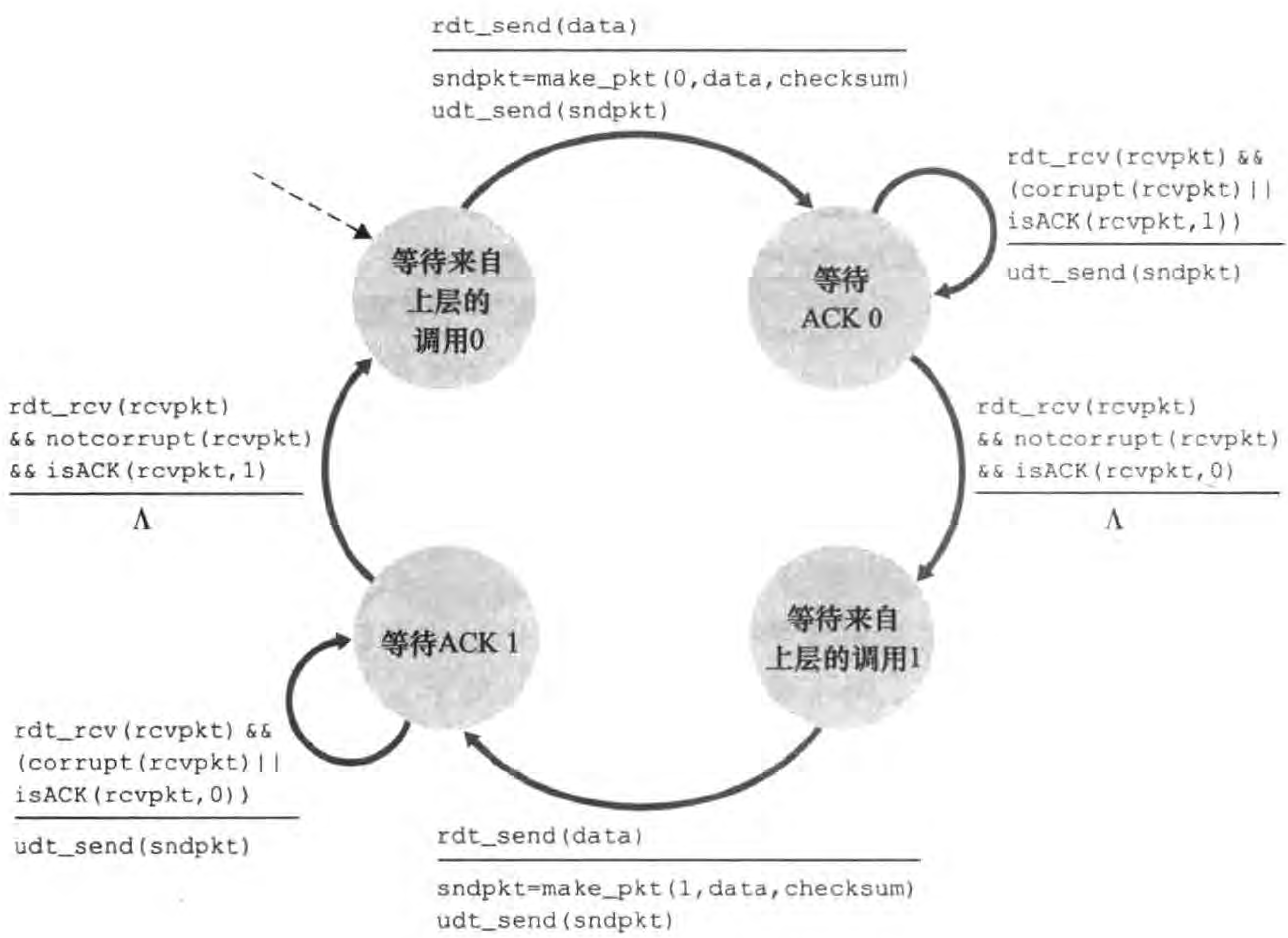


图 3-13 rdt2.2 发送方

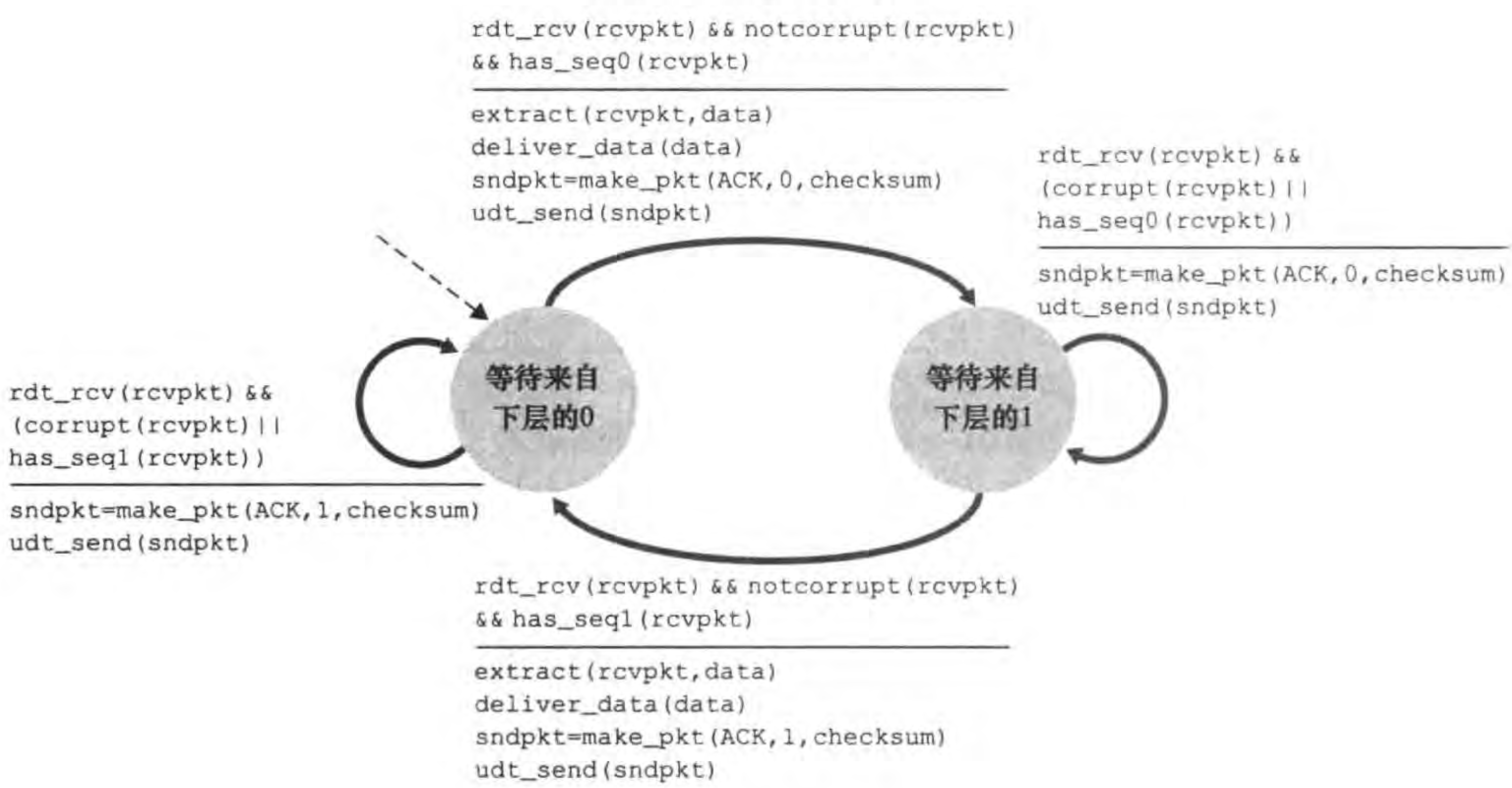


图 3-14 rdt2.2 接收方

3. 经具有比特差错的丢包信道的可靠数据传输：rdt3.0

现在假定除了比特受损外，底层信道还会丢包，这在今天的计算机网络（包括因特网）中并不罕见。协议现在必须处理另外两个关注的问题：怎样检测丢包以及发生丢包后该做些什么。在 rdt2.2 中已经研发的技术，如使用检验和、序号、ACK 分组和重传等，使我们能给出后一个问题的答案。为解决第一个关注的问题，还需增加一种新的协议机制。

有很多可能的方法用于解决丢包问题（在本章结尾的习题中研究了几种其他方法）。

这里，我们让发送方负责检测和恢复丢包工作。假定发送方传输一个数据分组，该分组或者接收方对该分组的 ACK 发生了丢失。在这两种情况下，发送方都收不到应当到来的接收方的响应。如果发送方愿意等待足够长的时间以便确定分组已丢失，则它只需重传该数据分组即可。你应该相信该协议确实有效。

但是发送方需要等待多久才能确定已丢失了某些东西呢？很明显发送方至少需要等待这样长的时间：即发送方与接收方之间的一个往返时延（可能会包括在中间路由器的缓冲时延）加上接收方处理一个分组所需的时间。在很多网络中，最坏情况下的最大时延是很难估算的，确定的因素非常少。此外，理想的协议应尽可能快地从丢包中恢复出来；等待一个最坏情况的时延可能意味着要等待一段较长的时间，直到启动差错恢复为止。因此实践中采取的方法是发送方明智地选择一个时间值，以判定可能发生了丢包（尽管不能确保）。如果在这个时间内没有收到 ACK，则重传该分组。注意到如果一个分组经历了一个特别大的时延，发送方可能会重传该分组，即使该数据分组及其 ACK 都没有丢失。这就在发送方到接收方的信道中引入了**冗余数据分组**（duplicate data packet）的可能性。幸运的是，rdt2.2 协议已经有足够的功能（即序号）来处理冗余分组情况。

从发送方的观点来看，重传是一种万能灵药。发送方不知道是一个数据分组丢失，还是一个 ACK 丢失，或者只是该分组或 ACK 过度延时。在所有这些情况下，动作是同样的：重传。为了实现基于时间的重传机制，需要一个**倒计数定时器**（countdown timer），在一个给定的时间量过期后，可中断发送方。因此，发送方需要能做到：①每次发送一个分组（包括第一次分组和重传分组）时，便启动一个定时器。②响应定时器中断（采取适当的动作）。③终止定时器。

图 3-15 给出了 rdt3.0 的发送方 FSM，这是一个在可能出错和丢包的信道上可靠传

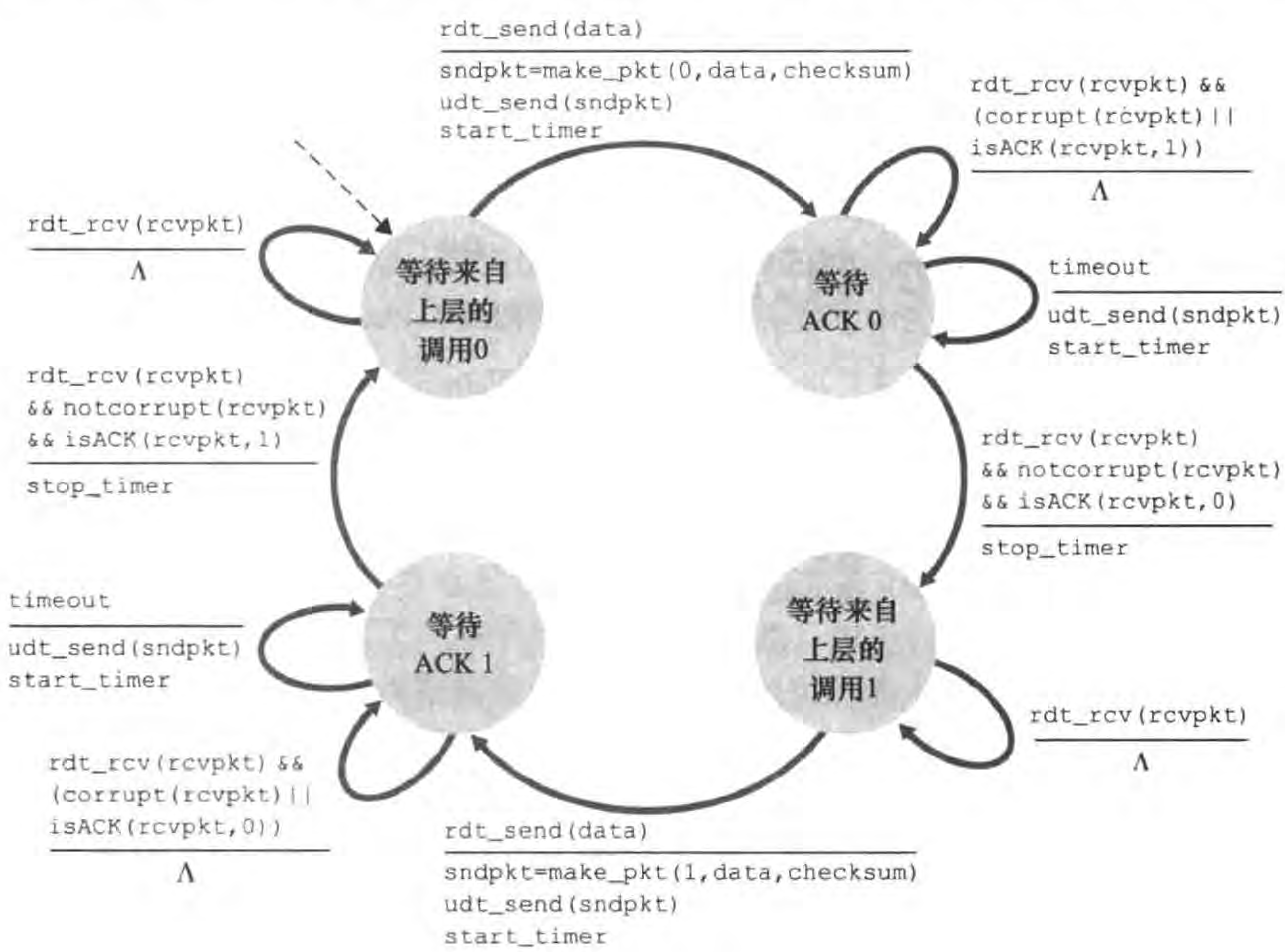


图 3-15 rdt3.0 发送方

输数据的协议；在课后习题中，将请你提供 rdt3.0 的接收方 FSM。图 3-16 显示了在没有丢包和延迟分组情况下协议运作的情况，以及它是如何处理数据分组丢失的。在图 3-16 中，时间从图的顶部朝底部移动；注意到一个分组的接收时间必定迟于一个分组的发送时间，这是因为发送时延与传播时延之故。在图 3-16b ~ d 中，发送方括号部分表明了定时器的设置时刻以及随后的超时。本章后面的习题探讨了该协议几个更细微的方面。因为分组序号在 0 和 1 之间交替，因此 rdt3.0 有时被称为**比特交替协议**（alternating-bit protocol）。

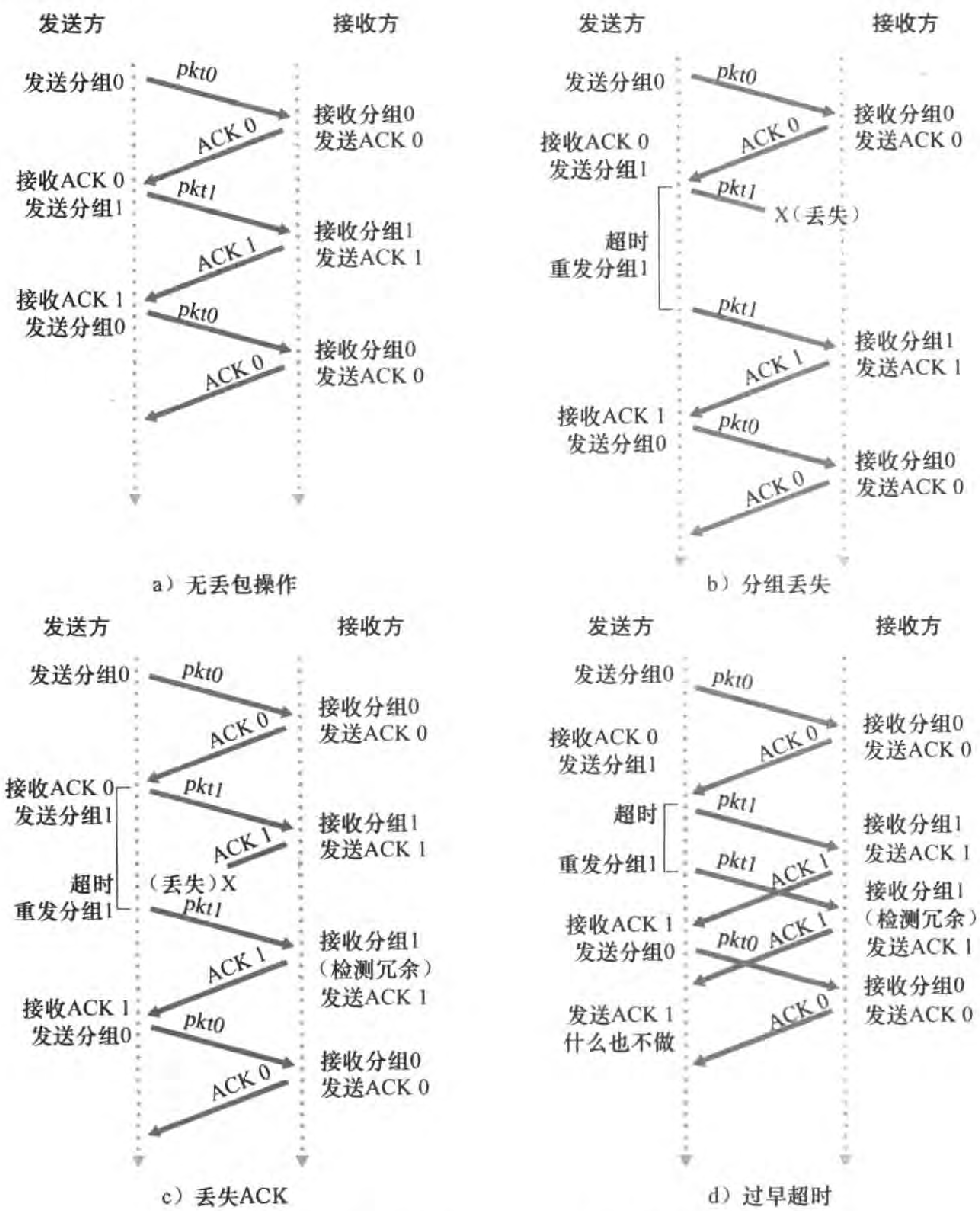


图 3-16　rdt3.0 的运行，比特交替协议

现在我们归纳一下数据传输协议的要点。在检验和、序号、定时器、肯定和否定确认分组这些技术中，每种机制都在协议的运行中起到了必不可少的作用。至此，我们得到了一个可靠数据传输协议！

3.4.2 流水线可靠数据传输协议

rdt3.0 是一个功能正确的协议，但并非人人都对它的性能满意，特别是在今天的高速网络中更是如此。rdt3.0 性能问题的核心在于它是一个停等协议。

为了评价该停等行为对性能的影响，可考虑一种具有两台主机的理想化场合，一台主机位于美国西海岸，另一台位于美国东海岸，如图 3-17 所示。在这两个端系统之间的光速往返传播时延 RTT 大约为 30 毫秒。假定彼此通过一条发送速率 R 为 1Gbps（每秒 10^9 比特）的信道相连。包括首部字段和数据的分组长 L 为 1000 字节（8000 比特），发送一个分组进入 1Gbps 链路实际所需时间是：

$$t_{\text{trans}} = \frac{L}{R} = \frac{8000\text{bit/pkt}}{10^9\text{bit/s}} = 8\mu\text{s/pkt}$$

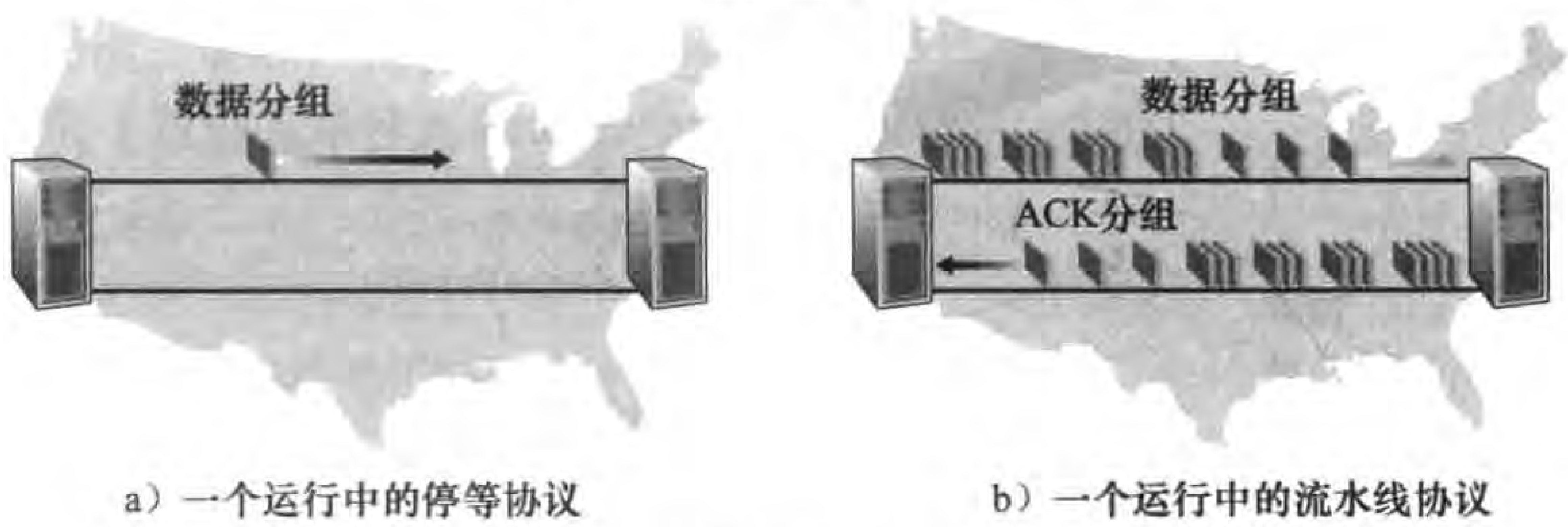


图 3-17 停等协议与流水线协议

图 3-18a 显示了对于该停等协议，如果发送方在 $t=0$ 时刻开始发送分组，则在 $t=L/R=8\mu\text{s}$ 后，最后 1 比特数据进入了发送端信道。该分组经过 15ms 的穿越国家的旅途后到达接收端，该分组的最后 1 比特在时刻 $t=\text{RTT}/2+L/R=15.008\text{ms}$ 时到达接收方。为了简化起见，假设 ACK 分组很小（以便我们可以忽略其发送时间），接收方一旦收到一个数据分组的最后 1 比特后立即发送 ACK，ACK 在时刻 $t=\text{RTT}+L/R=30.008\text{ms}$ 时在发送方出现。此时，发送方可以发送下一个报文。因此，在 30.008ms 内，发送方的发送只用了 0.008ms。如果我们定义发送方（或信道）的**利用率**（utilization）为：发送方实际忙于将发送比特送进信道的那部分时间与发送时间之比，图 3-18a 中的分析表明了停等协议有着非常低的发送方利用率 U_{sender}：

$$U_{\text{sender}} = \frac{L/R}{\text{RTT}+L/R} = \frac{0.008}{30.008} = 0.000\,27$$

这就是说，发送方只有万分之 2.7 时间是忙的。从其他角度来看，发送方在 30.008ms 内只能发送 1000 字节，有效的吞吐量仅为 267kbps，即使有 1Gbps 的链路可用也是如此！想象一个不幸的网络经理购买了一条千兆比容量的链路，但他仅能得到 267kbps 吞吐量的情况！这是一个形象的网络协议限制底层网络硬件所提供的能力的图例。而且，我们还忽略了在发送方和接收方的底层协议处理时间，以及可能出现在发送方与接收方之间的任何中间路由器上的处理与排队时延。考虑到这些因素，将进一步增加时延，使其性能更糟糕。

这种特殊的性能问题的一个简单解决方法是：不以停等方式运行，允许发送方发送多个分组而无须等待确认，如在图 3-17b 图示的那样。图 3-18b 显示了如果发送方可以在等待确认之前发送 3 个报文，其利用率也基本上提高 3 倍。因为许多从发送方向接收方输送

的分组可以被看成是填充到一条流水线中，故这种技术被称为**流水线**（pipelining）。流水线技术对可靠数据传输协议可带来如下影响：

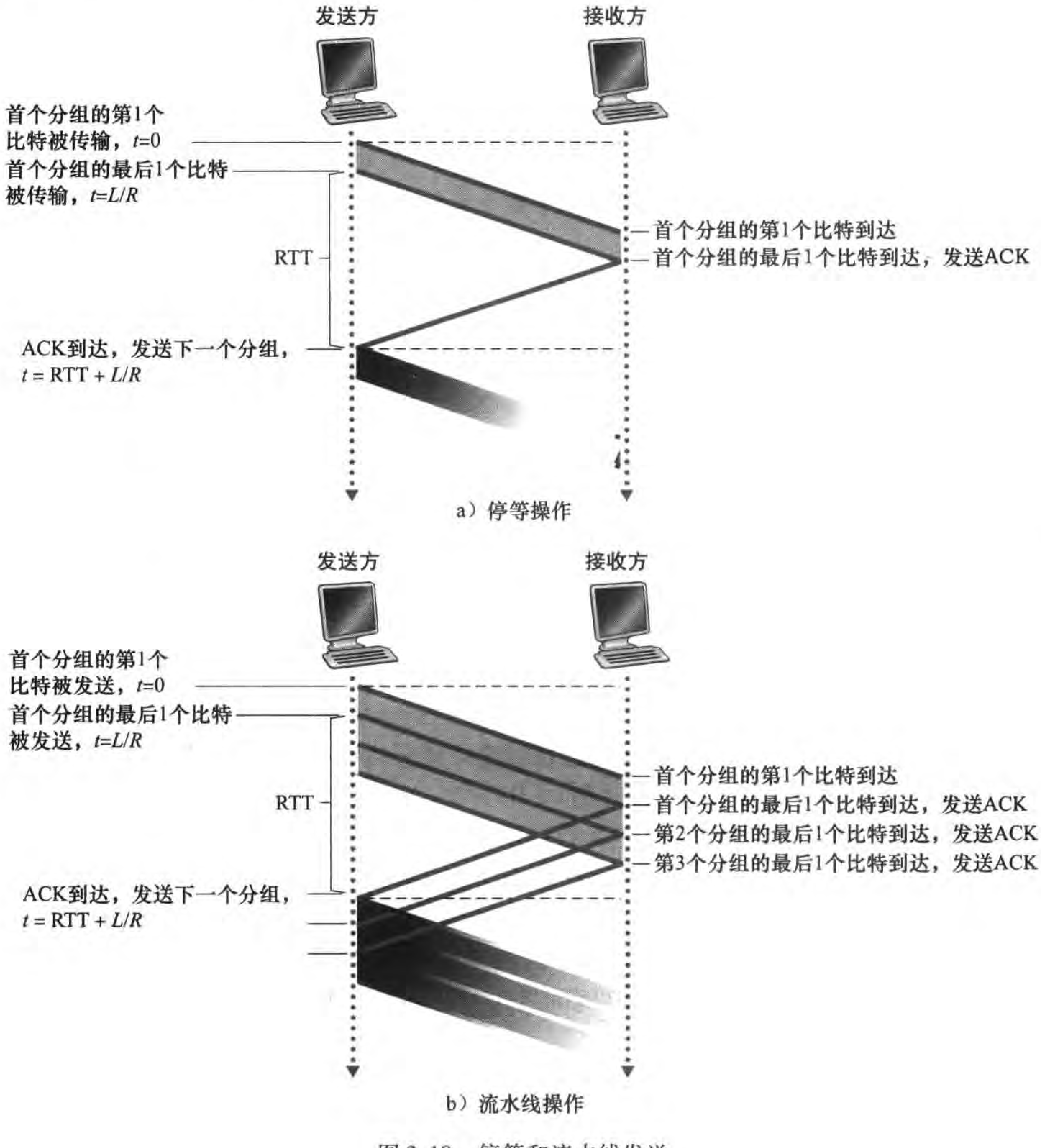


图3-18　停等和流水线发送

- 必须增加序号范围，因为每个输送中的分组（不计算重传的）必须有一个唯一的序号，而且也许有多个在输送中的未确认报文。
- 协议的发送方和接收方两端也许不得不缓存多个分组。发送方最低限度应当能缓冲那些已发送但没有确认的分组。如下面讨论的那样，接收方或许也需要缓存那些已正确接收的分组。
- 所需序号范围和对缓冲的要求取决于数据传输协议如何处理丢失、损坏及延时过大的分组。解决流水线的差错恢复有两种基本方法是：**回退 *N* 步**（Go-Back-N，GBN）和**选择重传**（Selective Repeat，SR）。

3.4.3 回退 N 步

在**回退 N 步**（GBN）**协议**中，允许发送方发送多个分组（当有多个分组可用时）而不需等待确认，但它也受限于在流水线中未确认的分组数不能超过某个最大允许数 N。在本节中我们较为详细地描述 GBN。但在继续阅读之前，建议你操作本书配套 Web 网站上的 GBN Java 小程序（这是一个非常好的 Java 程序）。

图 3-19 显示了发送方看到的 GBN 协议的序号范围。如果我们将基序号（base）定义为最早未确认分组的序号，将下一个序号（nextseqnum）定义为最小的未使用序号（即下一个待发分组的序号），则可将序号范围分割成 4 段。在 [0, base - 1] 段内的序号对应于已经发送并被确认的分组。[base, nextseqnum - 1] 段内对应已经发送但未被确认的分组。[nextseqnum, base + N - 1] 段内的序号能用于那些要被立即发送的分组，如果有数据来自上层的话。最后，大于或等于 base + N 的序号是不能使用的，直到当前流水线中未被确认的分组（特别是序号为 base 的分组）已得到确认为止。

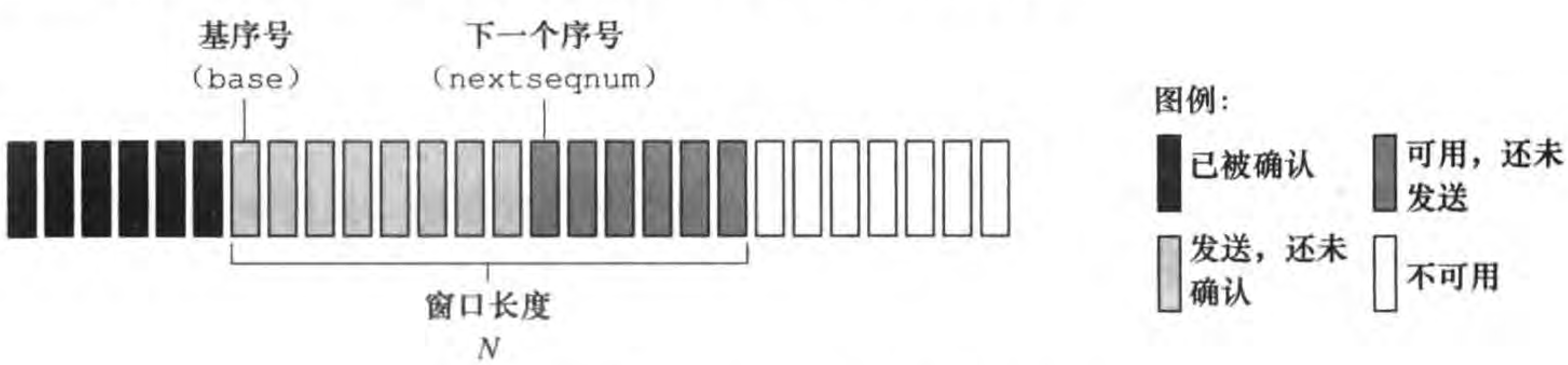


图 3-19 在 GBN 中发送方看到的序号

如图 3-19 所提示的那样，那些已被发送但还未被确认的分组的许可序号范围可以被看成是一个在序号范围内长度为 N 的窗口。随着协议的运行，该窗口在序号空间向前滑动。因此，N 常被称为**窗口长度**（window size），GBN 协议也常被称为**滑动窗口协议**（sliding-window protocol）。你也许想知道，我们为什么先要限制这些被发送的、未被确认的分组的数目为 N 呢？为什么不允许这些分组为无限制的数目呢？我们将在 3.5 节看到，流量控制是对发送方施加限制的原因之一。我们将在 3.7 节学习 TCP 拥塞控制时分析另一个原因。

在实践中，一个分组的序号承载在分组首部的一个固定长度的字段中。如果分组序号字段的比特数是 k，则该序号范围是 $[0, 2^k - 1]$。在一个有限的序号范围内，所有涉及序号的运算必须使用模 2^k 运算。（即序号空间可被看作是一个长度为 2^k 的环，其中序号 $2^k - 1$ 紧接着序号 0。）前面讲过，rdt3.0 有一个 1 比特的序号，序号范围是 [0, 1]。在本章末的几道习题中探讨了一个有限的序号范围所产生的结果。我们将在 3.5 节看到，TCP 有一个 32 比特的序号字段，其中的 TCP 序号是按字节流中的字节进行计数的，而不是按分组计数。

图 3-20 和图 3-21 给出了一个基于 ACK、无 NAK 的 GBN 协议的发送方和接收方这两端的扩展 FSM 描述。我们称该 FSM 描述为*扩展 FSM*，是因为我们已经增加了变量（类似于编程语言中的变量）base 和 nextseqnum，还增加了对这些变量的操作以及与这些变量有关的条件动作。注意到该扩展的 FSM 规约现在变得有点像编程语言规约。[Bochman 1984] 对 FSM 扩展技术提供了一个很好的综述，也提供了用于定义协议的其他基于编程语言的技术。

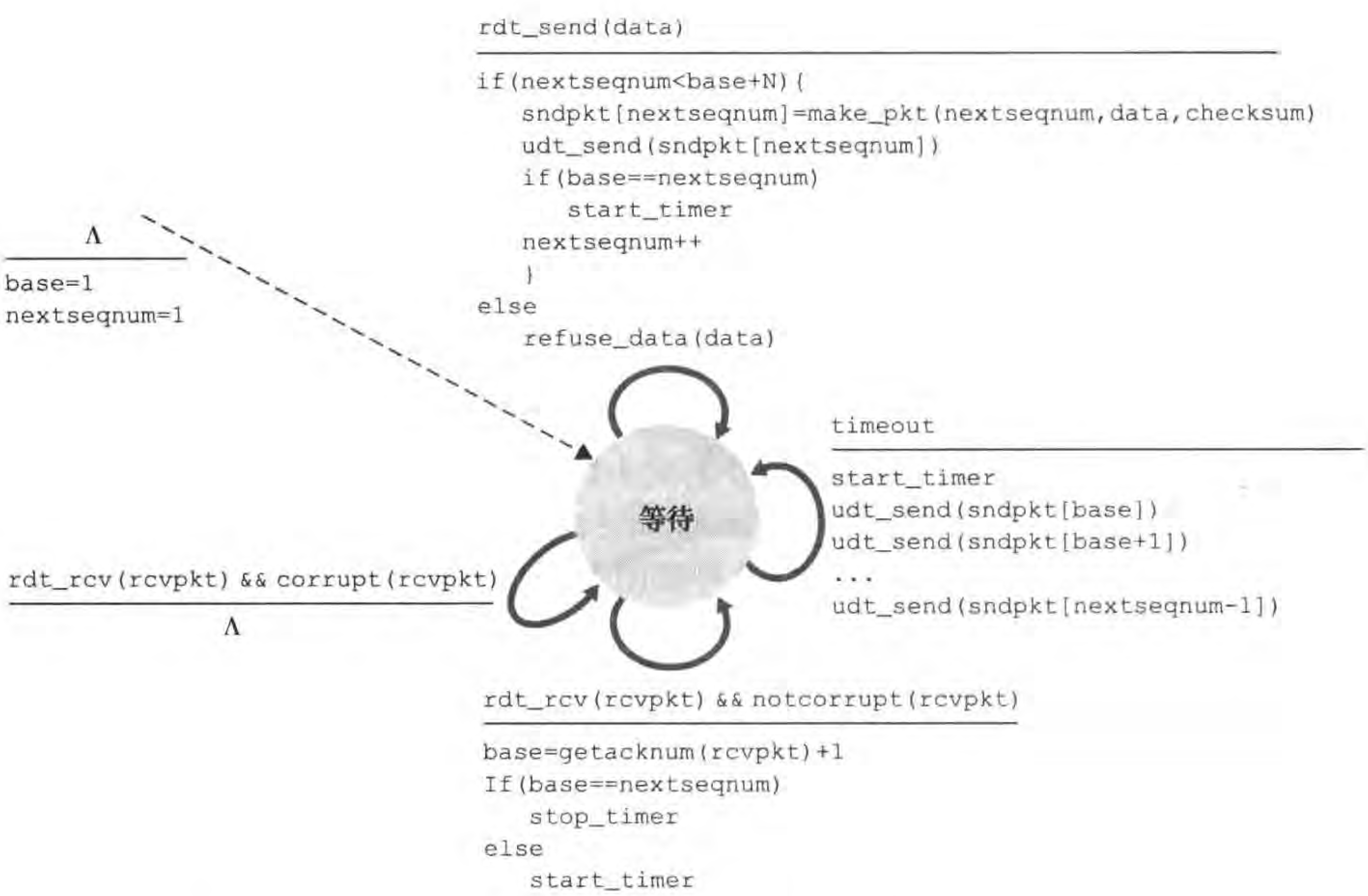


图 3-20 GBN 发送方的扩展 FSM 描述

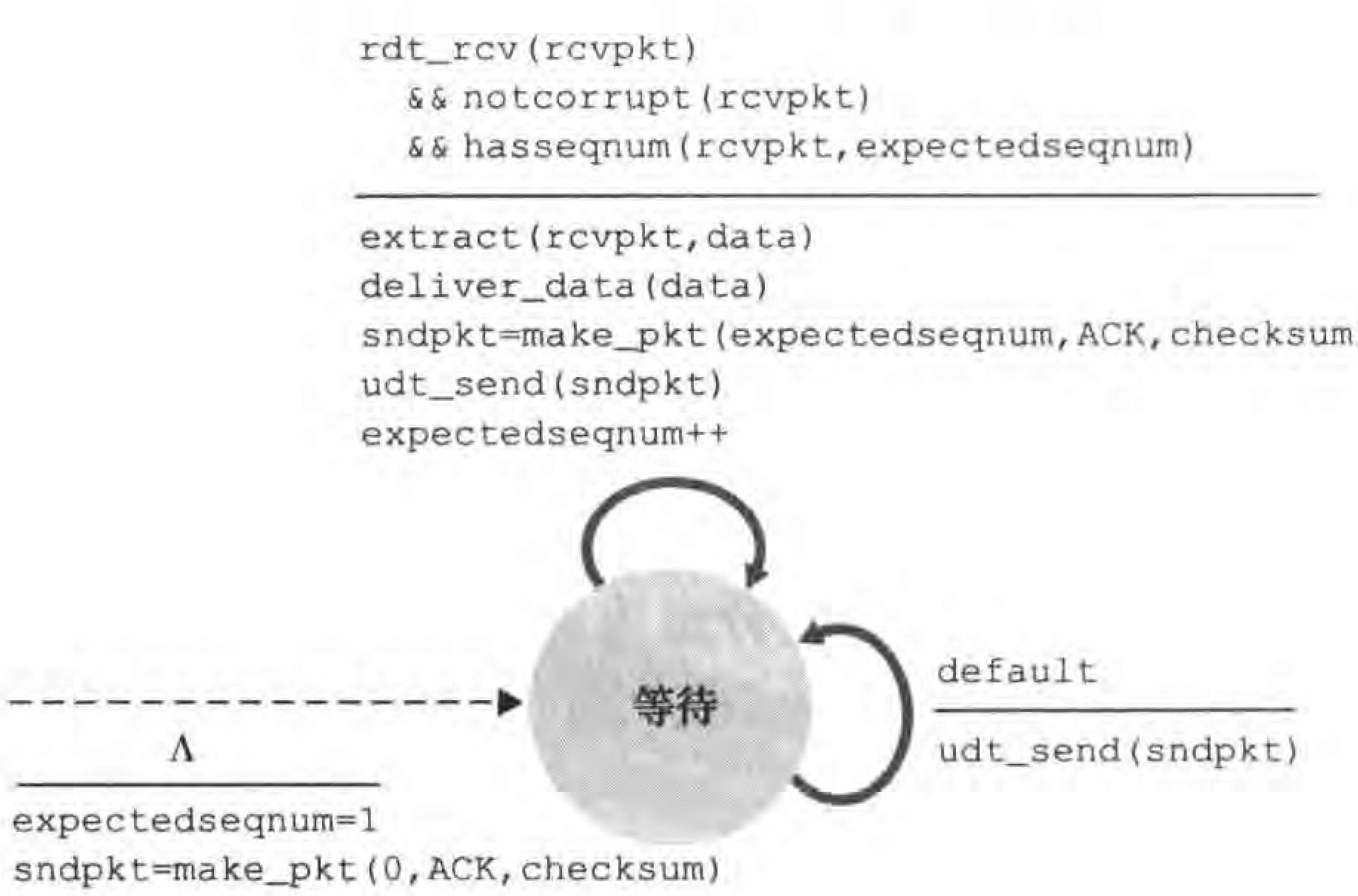


图 3-21 GBN 接收方的扩展 FSM 描述

GBN 发送方必须响应三种类型的事件：

- 上层的调用。当上层调用 rdt_send()时，发送方首先检查发送窗口是否已满，即是否有 N 个已发送但未被确认的分组。如果窗口未满，则产生一个分组并将其发送，并相应地更新变量。如果窗口已满，发送方只需将数据返回给上层，隐式地指示上层该窗口已满。然后上层可能会过一会儿再试。在实际实现中，发送方更可能缓存（并不立刻发送）这些数据，或者使用同步机制（如一个信号量或标志）允许上层在仅当窗口不满时才调用 rdt_send()。
- 收到一个 ACK。在 GBN 协议中，对序号为 n 的分组的确认采取**累积确认**（cumulative acknowledgment）的方式，表明接收方已正确接收到序号为 n 的以前且包括 n 在内的所有分组。稍后讨论 GBN 接收方一端时，我们将再次研究这个主题。

- 超时事件。协议的名字"回退 N 步"来源于出现丢失和时延过长分组时发送方的行为。就像在停等协议中那样，定时器将再次用于恢复数据或确认分组的丢失。如果出现超时，发送方重传所有已发送但还未被确认过的分组。图 3-20 中的发送方仅使用一个定时器，它可被当作是最早的已发送但未被确认的分组所使用的定时器。如果收到一个 ACK，但仍有已发送但未被确认的分组，则定时器被重新启动。如果没有已发送但未被确认的分组，停止该定时器。

在 GBN 中，接收方的动作也很简单。如果一个序号为 n 的分组被正确接收到，并且按序（即上次交付给上层的数据是序号为 $n-1$ 的分组），则接收方为分组 n 发送一个 ACK，并将该分组中的数据部分交付到上层。在所有其他情况下，接收方丢弃该分组，并为最近按序接收的分组重新发送 ACK。注意到因为一次交付给上层一个分组，如果分组 k 已接收并交付，则所有序号比 k 小的分组也已经交付。因此，使用累积确认是 GBN 一个自然的选择。

在 GBN 协议中，接收方丢弃所有失序分组。尽管丢弃一个正确接收（但失序）的分组有点愚蠢和浪费，但这样做是有理由的。前面讲过，接收方必须按序将数据交付给上层。假定现在期望接收分组 n，而分组 $n+1$ 却到了。因为数据必须按序交付，接收方可能缓存（保存）分组 $n+1$，然后，在它收到并交付分组 n 后，再将该分组交付到上层。然而，如果分组 n 丢失，则该分组及分组 $n+1$ 最终将在发送方根据 GBN 重传规则而被重传。因此，接收方只需丢弃分组 $n+1$ 即可。这种方法的优点是接收缓存简单，即接收方不需要缓存任何失序分组。因此，虽然发送方必须维护窗口的上下边界及 nextseqnum 在该窗口中的位置，但是接收方需要维护的唯一信息就是下一个按序接收的分组的序号。该值保存在 expectedseqnum 变量中，如图 3-21 中接收方 FSM 所示。当然，丢弃一个正确接收的分组的缺点是随后对该分组的重传也许会丢失或出错，因此甚至需要更多的重传。

图 3-22 给出了窗口长度为 4 个分组的 GBN 协议的运行情况。因为该窗口长度的限制，发送方发送分组 0~3，然后在继续发送之前，必须等待直到一个或多个分组被确认。当接收到每一个连续的 ACK（例如 ACK 0 和 ACK 1）时，该窗口便向前滑动，发送方便可以发送新的分组（分别是分组 4 和分组 5）。在接收方，分组 2 丢失，因此分组 3、4 和 5 被发现是失序分组并被丢弃。

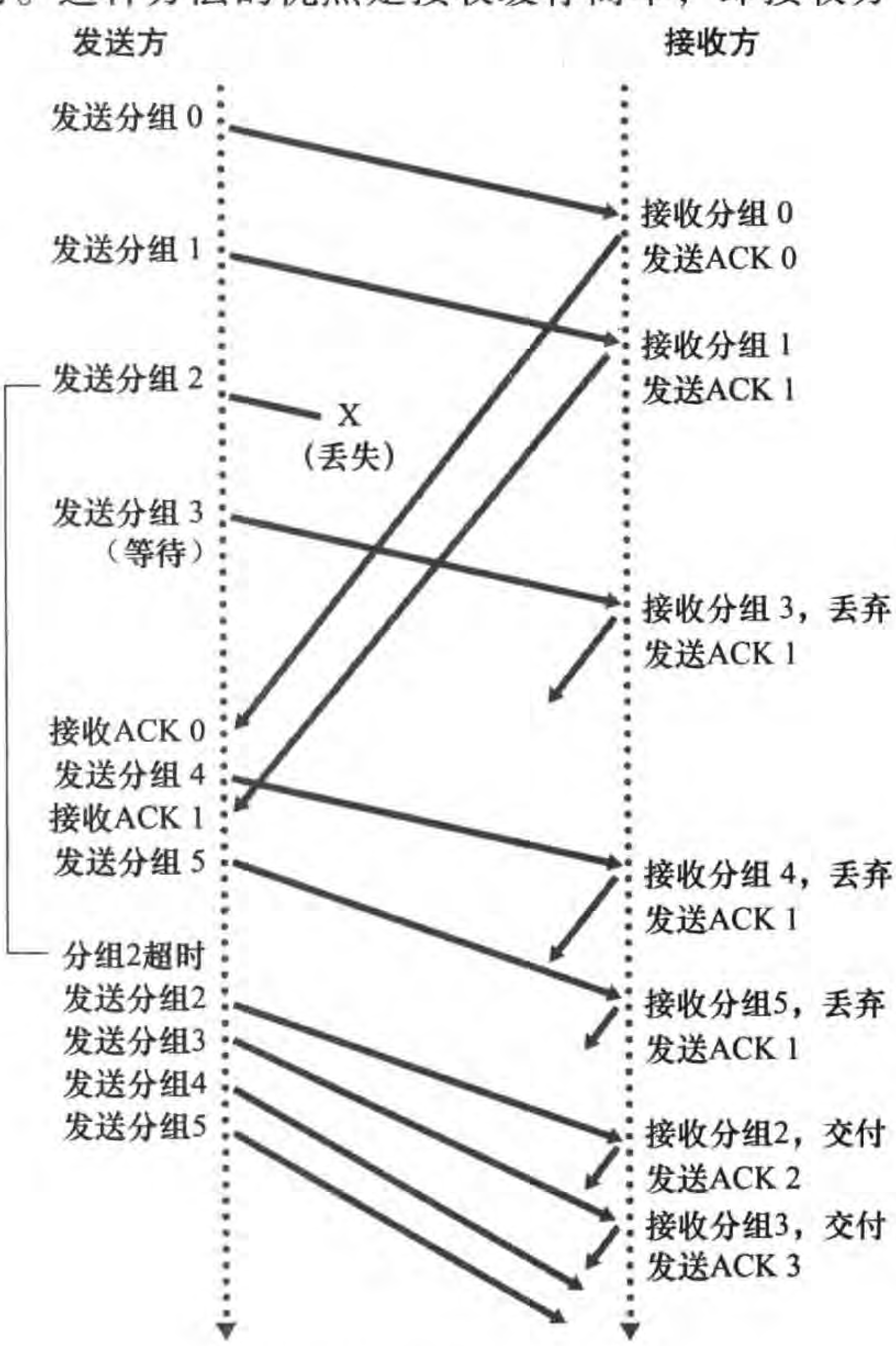


图 3-22 运行中的 GBN

在结束对 GBN 的讨论之前，需要提请注意的是，在协议栈中实现该协议可能与图 3-20 中的扩展 FSM 有相似的

结构。该实现也可能是以各种过程形式出现，每个过程实现了在响应各种可能出现的事件时要采取的动作。在这种**基于事件的编程**（event-based programming）方式中，这些过程要么被协议栈中的其他过程调用，要么作为一次中断的结果。在发送方，这些事件包括：①来自上层实体的调用去调用 rdt_send()；②定时器中断；③报文到达时，来自下层的调用去调用 rdt_rcv()。本章后面的编程作业会使你有机会在一个模拟网络环境中实际实现这些例程，但该环境却是真实的。

这里我们注意到，GBN 协议中综合了我们将在 3.5 节中学习 TCP 可靠数据传输构件时遇到的所有技术。这些技术包括使用序号、累积确认、检验和以及超时/重传操作。

3.4.4　选择重传

在图 3-17 中，GBN 协议潜在地允许发送方用多个分组“填充流水线”，因此避免了停等协议中所提到的信道利用率问题。然而，GBN 本身也有一些情况存在着性能问题。尤其是当窗口长度和带宽时延积都很大时，在流水线中会有很多分组更是如此。单个分组的差错就能够引起 GBN 重传大量分组，许多分组根本没有必要重传。随着信道差错率的增加，流水线可能会被这些不必要重传的分组所充斥。想象一下，在我们口述消息的例子中，如果每次有一个单词含糊不清，其前后 1000 个单词（例如，窗口长度为 1000 个单词）不得不被重传的情况。此次口述会由于这些反复述说的单词而变慢。

顾名思义，选择重传（SR）协议通过让发送方仅重传那些它怀疑在接收方出错（即丢失或受损）的分组而避免了不必要的重传。这种个别的、按需的重传要求接收方逐个地确认正确接收的分组。再次用窗口长度 N 来限制流水线中未完成、未被确认的分组数。然而，与 GBN 不同的是，发送方已经收到了对窗口中某些分组的 ACK。图 3-23 显示了 SR 发送方看到的序号空间。图 3-24 详细描述了 SR 发送方所采取的动作。

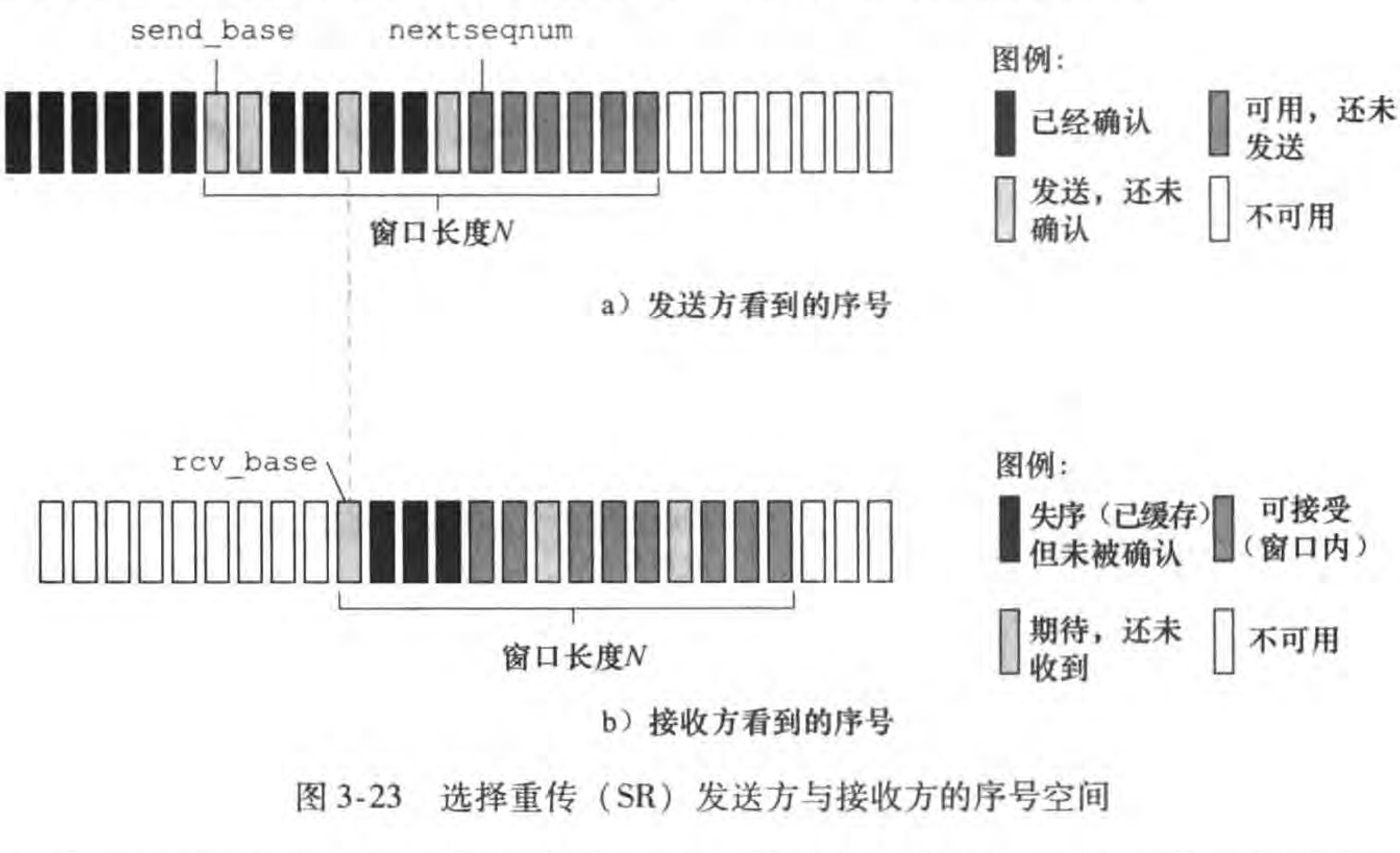


图 3-23　选择重传（SR）发送方与接收方的序号空间

SR 接收方将确认一个正确接收的分组而不管其是否按序。失序的分组将被缓存直到所有丢失分组（即序号更小的分组）皆被收到为止，这时才可以将一批分组按序交付给上层。图 3-25 详细列出了 SR 接收方所采用的各种动作。图 3-26 给出了一个例子以说明出现丢包时 SR 的操作。值得注意的是，在图 3-26 中接收方初始时缓存了分组 3、4、5，并

在最终收到分组 2 时，才将它们一并交付给上层。

1. *从上层收到数据*。当从上层接收到数据后，SR 发送方检查下一个可用于该分组的序号。如果序号位于发送方的窗口内，则将数据打包并发送；否则就像在 GBN 中一样，要么将数据缓存，要么将其返回给上层以便以后传输。

2. *超时*。定时器再次被用来防止丢失分组。然而，现在每个分组必须拥有其自己的逻辑定时器，因为超时发生后只能发送一个分组。可以使用单个硬件定时器模拟多个逻辑定时器的操作［Varghese 1997］。

3. *收到 ACK*。如果收到 ACK，倘若该分组序号在窗口内，则 SR 发送方将那个被确认的分组标记为已接收。如果该分组的序号等于 send_base，则窗口基序号向前移动到具有最小序号的未确认分组处。如果窗口移动了并且有序号落在窗口内的未发送分组，则发送这些分组。

图 3-24　SR 发送方的事件与动作

1. *序号在［rcv_base，rcv_base + N - 1］内的分组被正确接收*。在此情况下，收到的分组落在接收方的窗口内，一个选择 ACK 被回送给发送方。如果该分组以前没收到过，则缓存该分组。如果该分组的序号等于接收窗口的基序号（图 3-23 中的 rcv_base），则该分组以及以前缓存的序号连续的（起始于 rcv_base 的）分组交付给上层。然后，接收窗口按向前移动分组的编号向上交付这些分组。举例子来说，考虑一下图 3-26。当收到一个序号为 rcv_base = 2的分组时，该分组及分组 3、4、5 可被交付给上层。

2. *序号在［rcv_base - N，rcv_base - 1］内的分组被正确收到*。在此情况下，必须产生一个 ACK，即使该分组是接收方以前已确认过的分组。

3. *其他情况*。忽略该分组。

图 3-25　SR 接收方的事件与动作

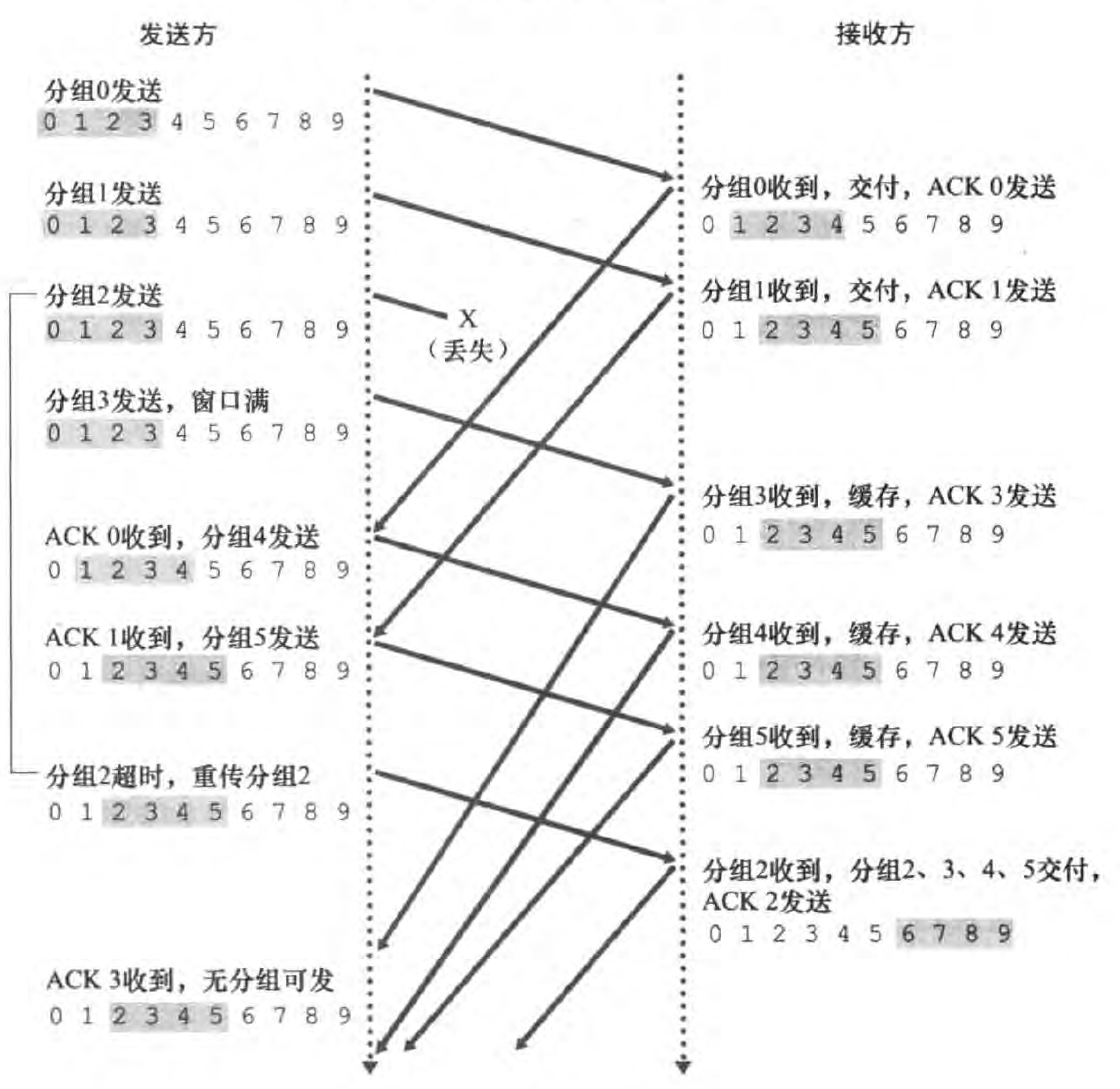


图 3-26　SR 操作

注意到图3-25中的第二步很重要，接收方重新确认（而不是忽略）已收到过的那些序号小于当前窗口基序号的分组。你应该理解这种重新确认确实是需要的。例如，给定在图3-23中所示的发送方和接收方的序号空间，如果分组send_base的ACK没有从接收方传播回发送方，则发送方最终将重传分组send_base，即使显然（对我们而不是对发送方来说！）接收方已经收到了该分组。如果接收方不确认该分组，则发送方窗口将永远不能向前滑动！这个例子说明了SR协议（和很多其他协议一样）的一个重要方面。对于哪些分组已经被正确接收，哪些没有，发送方和接收方并不总是能看到相同的结果。对SR协议而言，这就意味着发送方和接收方的窗口并不总是一致。

当我们面对有限序号范围的现实时，发送方和接收方窗口间缺乏同步会产生严重的后果。考虑下面例子中可能发生的情况，该例有包括4个分组序号0、1、2、3的有限序号范围且窗口长度为3。假定发送了分组0至2，并在接收方被正确接收且确认了。此时，接收方窗口落在第4、5、6个分组上，其序号分别为3、0、1。现在考虑两种情况。在第一种情况下，如图3-27a所示，对前3个分组的ACK丢失，因此发送方重传这些分组。因此，接收方下一步要接收序号为0的分组，即第一个发送分组的副本。

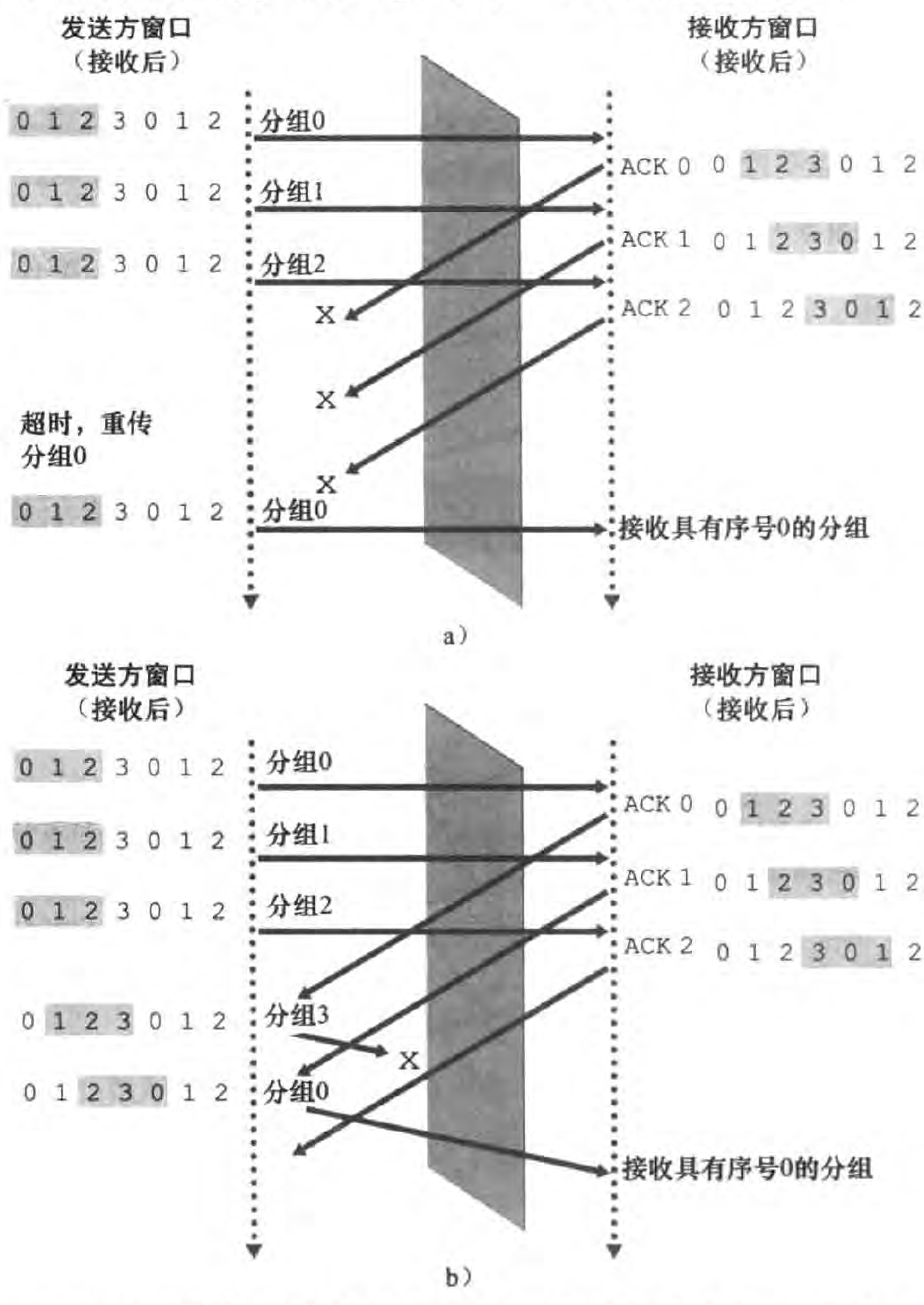


图3-27　SR接收方窗口太大的困境：是一个新分组还是一次重传

在第二种情况下，如图3-27b所示，对前3个分组的ACK都被正确交付。因此发送方向前移动窗口并发送第4、5、6个分组，其序号分别为3、0、1。序号为3的分组丢失，但序号为0的分组到达（一个包含新数据的分组）。

现在考虑一下图3-27中接收方的观点，在发送方和接收方之间有一个假想的帘子，因为接收方不能“看见”发送方采取的动作。接收方所能观察到的是它从信道中收到的以及它向信道中发出报文序列。就其所关注的而言，图3-27中的两种情况是等同的。没有办法区分是第1个分组的重传还是第5个分组的初次传输。显然，窗口长度比序号空间小1时协议无法工作。但窗口必须多小呢？本章后面的一道习题请你说明为何对于SR协议而言，窗口长度必须小于或等于序号空间大小的一半。

在本书配套的网站上，可以找到一个模仿SR协议运行的Java小程序。尝试进行你以前对GBN Java小程序所进行的相同的实验。这些结果与你期望的一致吗？

至此我们结束了对可靠数据传输协议的讨论。我们已涵盖许多基础知识，并介绍了多种机制，这些机制可一起提供可靠数据传输。表3-1总结这些机制。既然我们已经学习了所有这些运行中的机制，并能看到“全景”，我们建议你再复习一遍本节内容，看看这些机制是怎样逐步被添加进来，以涵盖复杂性渐增的（现实的）连接发送方与接收方的各种信道模型的，或者如何改善协议性能的。

表3-1 可靠数据传输机制及其用途的总结

机制	用途和说明
检验和	用于检测在一个传输分组中的比特错误
定时器	用于超时/重传一个分组，可能因为该分组（或其ACK）在信道中丢失了。由于当一个分组延时但未丢失（过早超时），或当一个分组已被接收方收到但从接收方到发送方的ACK丢失时，可能产生超时事件，所以接收方可能会收到一个分组的多个冗余副本
序号	用于为从发送方流向接收方的数据分组按顺序编号。所接收分组的序号间的空隙可使接收方检测出丢失的分组。具有相同序号的分组可使接收方检测出一个分组的冗余副本
确认	接收方用于告诉发送方一个分组或一组分组已被正确地接收到了。确认报文通常携带着被确认的分组或多个分组的序号。确认可以是逐个的或累积的，这取决于协议
否定确认	接收方用于告诉发送方某个分组未被正确地接收。否定确认报文通常携带着未被正确接收的分组的序号
窗口、流水线	发送方也许被限制仅发送那些序号落在一个指定范围内的分组。通过允许一次发送多个分组但未被确认，发送方的利用率可在停等操作模式的基础上得到增加。我们很快将会看到，窗口长度可根据接收方接收和缓存报文的能力、网络中的拥塞程度或两者情况来进行设置

我们通过考虑在底层信道模型中的一个遗留假设来结束对可靠数据传输协议的讨论。前面讲过，我们曾假定分组在发送方与接收方之间的信道中不能被重新排序。这在发送方与接收方由单段物理线路相连的情况下，通常是一个合理的假设。然而，当连接两端的“信道”是一个网络时，分组重新排序是可能会发生的。分组重新排序的一个表现就是，一个具有序号或确认号 x 的分组的旧副本可能会出现，即使发送方或接收方的窗口中都没有包含 x。对于分组重新排序，信道可被看成基本上是在缓存分组，并在将来任意时刻自然地释放出这些分组。由于序号可以被重新使用，那么必须小心，以免出现这样的冗余分组。实际应用中采用的方法是，确保一个序号不被重新使用，直到发送方“确信”任何先前发送的序号为 x 的分组都不再在网络中为止。通过假定一个分组在网络中的“存活”时间不会超过某个固定最大时间量来做到这一点。在高速网络的TCP扩展中，最长的分组寿命被假定为大约3分钟［RFC 1323］。［Sunshine 1978］描述了一种使用序号的方法，它能

够完全避免重新排序问题。

3.5 面向连接的运输：TCP

既然我们已经学习了可靠数据传输的基本原理，我们就可以转而学习 TCP 了。TCP 是因特网运输层的面向连接的可靠的运输协议。我们在本节中将看到，为了提供可靠数据传输，TCP 依赖于前一节所讨论的许多基本原理，其中包括差错检测、重传、累积确认、定时器以及用于序号和确认号的首部字段。TCP 定义在 RFC 793、RFC 1122、RFC 1323、RFC 2018 以及 RFC 2581 中。

3.5.1 TCP 连接

TCP 被称为是**面向连接的**（connection-oriented），这是因为在一个应用进程可以开始向另一个应用进程发送数据之前，这两个进程必须先相互"握手"，即它们必须相互发送某些预备报文段，以建立确保数据传输的参数。作为 TCP 连接建立的一部分，连接的双方都将初始化与 TCP 连接相关的许多 TCP 状态变量（其中的许多状态变量将在本节和 3.7 节中讨论）。

历史事件

Vinton Cerf 和 Robert Kahn 与 TCP/IP

在 20 世纪 70 年代早期，分组交换网开始飞速增长，而因特网的前身 ARPAnet 也只是当时众多分组交换网中的一个。这些网络都有它们各自的协议。两个研究人员 Vinton Cerf 和 Robert Kahn 认识到互联这些网络的重要性，发明了沟通网络的 TCP/IP 协议，该协议代表**传输控制协议/网际协议**（Transmission Control Protocol/Internet Protocol）。虽然 Cerf 和 Kahn 开始时把该协议看成是单一的实体，但是后来将它分成单独运行的两个部分：TCP 和 IP。Cerf 和 Kahn 在 1974 年 5 月的《IEEE Transactions on Communications Technology》杂志上发表了一篇关于 TCP/IP 的论文 [Cerf 1974]。

TCP/IP 协议是当今因特网的支柱性协议，但它的发明先于 PC、工作站、智能手机和平板电脑，先于以太网、电缆、DSL、WiFi 和其他接入网技术的激增，先于 Web、社交媒体和流式视频等。Cerf 和 Kahn 预见到了对于联网协议的需求，一方面为行将定义的应用提供广泛的支持，另一方面允许任何主机与链路层协议互操作。

2004 年，Cerf 和 Kahn 由于"联网方面的开创性工作（包括因特网的基本通信协议 TCP/IP 的设计和实现）以及联网方面富有才能的领导"而获得 ACM 图灵奖，该奖项被认为是"计算机界的诺贝尔奖"。

这种 TCP "连接"不是一条像在电路交换网络中的端到端 TDM 或 FDM 电路。相反，该"连接"是一条逻辑连接，其共同状态仅保留在两个通信端系统的 TCP 程序中。前面讲过，由于 TCP 协议只在端系统中运行，而不在中间的网络元素（路由器和链路层交换机）中运行，所以中间的网络元素不会维持 TCP 连接状态。事实上，中间路由器对 TCP 连接完全视而不见，它们看到的是数据报，而不是连接。

台主机上的进程 B 存在一条 TCP 连接，那么应用层数据就可在从进程 B 流向进程 A 的同时，也从进程 A 流向进程 B。TCP 连接也总是**点对点**（point-to-point）的，即在单个发送方与单个接收方之间的连接。所谓“多播”（参见本书的在线补充材料），即在一次发送操作中，从一个发送方将数据传送给多个接收方，这种情况对 TCP 来说是不可能的。对于 TCP 而言，两台主机是一对，而 3 台主机则太多了！

我们现在来看看 TCP 连接是怎样建立的。假设运行在某台主机上的一个进程想与另一台主机上的一个进程建立一条连接。前面讲过，发起连接的这个进程被称为*客户进程*，而另一个进程被称为*服务器进程*。该客户应用进程首先要通知客户运输层，它想与服务器上的一个进程建立一条连接。2.7.2 节讲过，一个 Python 客户程序通过发出下面的命令来实现此目的。

```
clientSocket.connect((serverName,serverPort))
```

其中 serverName 是服务器的名字，serverPort 标识了服务器上的进程。客户上的 TCP 便开始与服务器上的 TCP 建立一条 TCP 连接。我们将在本节后面更为详细地讨论连接建立的过程。现在知道下列事实就可以了：客户首先发送一个特殊的 TCP 报文段，服务器用另一个特殊的 TCP 报文段来响应，最后，客户再用第三个特殊报文段作为响应。前两个报文段不承载“有效载荷”，也就是不包含应用层数据；而第三个报文段可以承载有效载荷。由于在这两台主机之间发送了 3 个报文段，所以这种连接建立过程常被称为**三次握手**（three-way handshake）。

一旦建立起一条 TCP 连接，两个应用进程之间就可以相互发送数据了。我们考虑一下从客户进程向服务器进程发送数据的情况。如 2.7 节中所述，客户进程通过套接字（该进程之门）传递数据流。数据一旦通过该门，它就由客户中运行的 TCP 控制了。如图 3-28 所示，TCP 将这些数据引导到该连接的**发送缓存**（send buffer）里，发送缓存是发起三次握手期间设置的缓存之一。接下来 TCP 就会不时从发送缓存里取出一块数据，并将数据传递到网络层。有趣的是，在 TCP 规范［RFC 793］中却没提及 TCP 应何时实际发送缓存里的数据，只是描述为“TCP 应该在它方便的时候以报文段的形式发送数据”。TCP 可从缓存中取出并放入报文段中的数据数量受限于**最大报文段长度**（Maximum Segment Size，MSS）。MSS 通常根据最初确定的由本地发送主机发送的最大链路层帧长度（即所谓的**最大传输单元**（Maximum Transmission Unit，MTU））来设置。设置该 MSS 要保证一个 TCP 报文段（当封装在一个 IP 数据报中）加上 TCP/IP 首部长度（通常 40 字节）将适合单个链路层帧。以太网和 PPP 链路层协议都具有 1500 字节的 MTU，因此 MSS 的典型值为 1460 字节。已经提出了多种发现路径 MTU 的方法，并基于路径 MTU 值设置 MSS（路径 MTU 是指能在从源到目的地的所有链路上发送的最大链路层帧［RFC 1191］）。注意到 MSS 是指在报文段里应用层数据的最大长度，而不是指包括首部的 TCP 报文段的最大长度。（该术语很容易混淆，但是我们不得不采用它，因为它已经根深蒂固了。）

TCP 为每块客户数据配上一个 TCP 首部，从而形成多个 **TCP 报文段**（TCP segment）。这些报文段被下传给网络层，网络层将其分别封装在网络层 IP 数据报中。然后这些 IP 数据报被发送到网络中。当 TCP 在另一端接收到一个报文段后，该报文段的数据就被放入该 TCP 连接的接收缓存中，如图 3-28 中所示。应用程序从此缓存中读取数据流。该连接的每一端都有各自的发送缓存和接收缓存。（读者可以参见 http://www.awl.com/kurose-ross 处的在线流控制 Java 小程序，它提供了关于发送缓存和接收缓存的一个动画演示。）

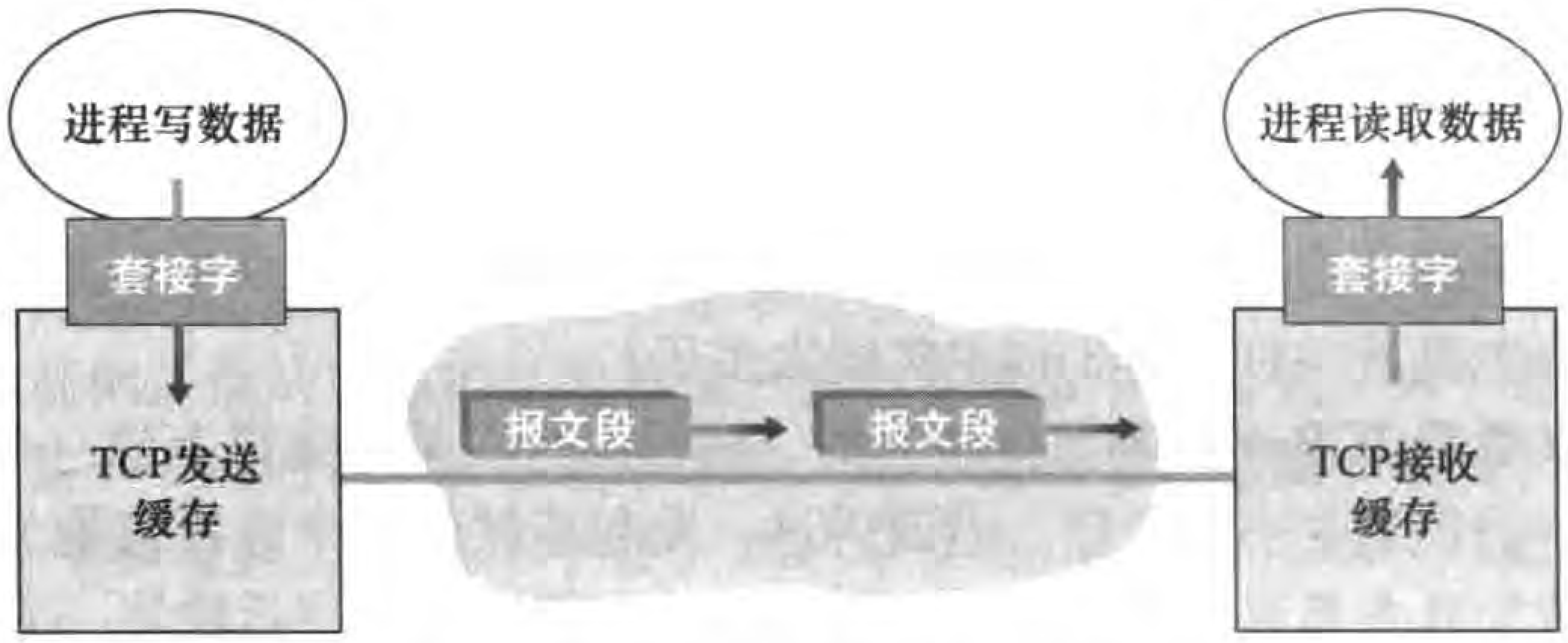


图3-28　TCP发送缓存和接收缓存

从以上讨论中我们可以看出，TCP连接的组成包括：一台主机上的缓存、变量和与进程连接的套接字，以及另一台主机上的另一组缓存、变量和与进程连接的套接字。如前面讲过的那样，在这两台主机之间的网络元素（路由器、交换机和中继器）中，没有为该连接分配任何缓存和变量。

3.5.2　TCP报文段结构

简要地了解了TCP连接后，我们研究一下TCP报文段结构。TCP报文段由首部字段和一个数据字段组成。数据字段包含一块应用数据。如前所述，MSS限制了报文段数据字段的最大长度。当TCP发送一个大文件，例如某Web页面上的一个图像时，TCP通常是将该文件划分成长度为MSS的若干块（最后一块除外，它通常小于MSS）。然而，交互式应用通常传送长度小于MSS的数据块。例如，对于像Telnet这样的远程登录应用，其TCP报文段的数据字段经常只有一个字节。由于TCP的首部一般是20字节（比UDP首部多12字节），所以Telnet发送的报文段也许只有21字节长。



图3-29　TCP报文段结构

图3-29显示了TCP报文段的结构。与UDP一样，首部包括**源端口号**和**目的端口号**，它被用于多路复用/分解来自或送到上层应用的数据。另外，同UDP一样，TCP首部也包括**检验和字段**（checksum field）。TCP报文段首部还包含下列字段：

- 32比特的**序号字段**（sequence number field）和32比特的**确认号字段**（acknowledgment number field）。这些字段被TCP发送方和接收方用来实现可靠数据传输服务，讨论见后。
- 16比特的**接收窗口字段**（receive window field），该字段用于流量控制。我们很快就会看到，该字段用于指示接收方愿意接受的字节数量。
- 4比特的**首部长度字段**（header length field），该字段指示了以32比特的字为单位的TCP首部长度。由于TCP选项字段的原因，TCP首部的长度是可变的。（通常，选项字段为空，所以TCP首部的典型长度是20字节。）

- 可选与变长的**选项字段**（options field），该字段用于发送方与接收方协商最大报文段长度（MSS）时，或在高速网络环境下用作窗口调节因子时使用。首部字段中还定义了一个时间戳选项。可参见 RFC 854 和 RFC 1323 了解其他细节。
- 6 比特的**标志字段**（flag field）。ACK 比特用于指示确认字段中的值是有效的，即该报文段包括一个对已被成功接收报文段的确认。RST、SYN 和 FIN 比特用于连接建立和拆除，我们将在本节后面讨论该问题。在明确拥塞通告中使用了 CWR 和 ECE 比特，如 3.7.2 节中讨论的那样。当 PSH 比特被置位时，就指示接收方应立即将数据交给上层。最后，URG 比特用来指示报文段里存在着被发送端的上层实体置为“紧急”的数据。紧急数据的最后一个字节由 16 比特的**紧急数据指针字段**（urgent data pointer field）指出。当紧急数据存在并给出指向紧急数据尾指针的时候，TCP 必须通知接收端的上层实体。（在实践中，PSH、URG 和紧急数据指针并没有使用。为了完整性起见，我们才提到这些字段。）

作为教师的经验是，学生有时觉得分组格式的讨论相当枯燥，也许有些乏味。特别是如果你和我们一样都喜爱乐高玩具，有关 TCP 首部的有趣和新颖的讨论请参见［Pomeranz 2010］。

1. 序号和确认号

TCP 报文段首部中两个最重要的字段是序号字段和确认号字段。这两个字段是 TCP 可靠传输服务的关键部分。但是在讨论这两个字段是如何用于提供可靠数据传输之前，我们首先来解释一下 TCP 在这两个字段中究竟放置了什么。

TCP 把数据看成一个无结构的、有序的字节流。我们从 TCP 对序号的使用上可以看出这一点，因为序号是建立在传送的字节流之上，而不是建立在传送的报文段的序列之上。**一个报文段的序号**（sequence number for a segment）因此是该报文段首字节的字节流编号。举例来说，假设主机 A 上的一个进程想通过一条 TCP 连接向主机 B 上的一个进程发送一个数据流。主机 A 中的 TCP 将隐式地对数据流中的每一个字节编号。假定数据流由一个包含 500 000 字节的文件组成，其 MSS 为 1000 字节，数据流的首字节编号是 0。如图 3-30 所示，该 TCP 将为该数据流构建 500 个报文段。给第一个报文段分配序号 0，第二个报文段分配序号 1000，第三个报文段分配序号 2000，以此类推。每一个序号被填入到相应 TCP 报文段首部的序号字段中。

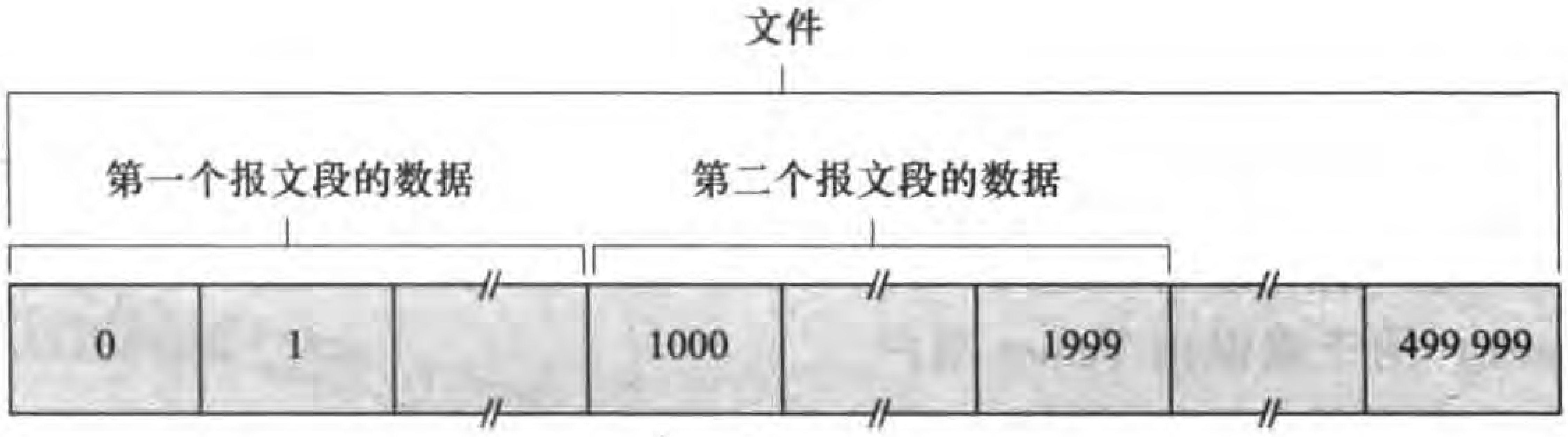


图 3-30 文件数据划分成 TCP 报文段

现在我们考虑一下确认号。确认号要比序号难处理一些。前面讲过，TCP 是全双工的，因此主机 A 在向主机 B 发送数据的同时，也许也接收来自主机 B 的数据（都是同一条 TCP 连接的一部分）。从主机 B 到达的每个报文段中都有一个序号用于从 B 流向 A 的数据。主机 A 填充进报文段的确认号是主机 A 期望从主机 B 收到的下一字节的序号。看一些例子有助于理解实际发生的事情。假设主机 A 已收到了来自主机 B 的编号为 0 ~ 535 的所有字节，同时

假设它打算发送一个报文段给主机 B。主机 A 等待主机 B 的数据流中字节 536 及之后的所有字节。所以主机 A 就会在它发往主机 B 的报文段的确认号字段中填上 536。

再举一个例子，假设主机 A 已收到一个来自主机 B 的包含字节 0 ~ 535 的报文段，以及另一个包含字节 900 ~ 1000 的报文段。由于某种原因，主机 A 还没有收到字节 536 ~ 899 的报文段。在这个例子中，主机 A 为了重新构建主机 B 的数据流，仍在等待字节 536（和其后的字节)。因此，A 到 B 的下一个报文段将在确认号字段中包含 536。因为 TCP 只确认该流中至第一个丢失字节为止的字节，所以 TCP 被称为提供**累积确认**（cumulative acknowledgment)。

最后一个例子也会引发一个重要而微妙的问题。主机 A 在收到第二个报文段（字节 536 ~ 899）之前收到第三个报文段（字节 900 ~ 1000)。因此，第三个报文段失序到达。该微妙的问题是：当主机在一条 TCP 连接中收到失序报文段时该怎么办？有趣的是，TCP RFC 并没有为此明确规定任何规则，而是把这一问题留给实现 TCP 的编程人员去处理。他们有两个基本的选择：①接收方立即丢弃失序报文段（如前所述，这可以简化接收方的设计)；②接收方保留失序的字节，并等待缺少的字节以填补该间隔。显然，后一种选择对网络带宽而言更为有效，是实践中采用的方法。

在图 3-30 中，我们假设初始序号为 0。事实上，一条 TCP 连接的双方均可随机地选择初始序号。这样做可以减少将那些仍在网络中存在的来自两台主机之间先前已终止的连接的报文段，误认为是后来这两台主机之间新建连接所产生的有效报文段的可能性（它碰巧与旧连接使用了相同的端口号）[Sunshine 1978]。

2. Telnet：序号和确认号的一个学习案例

Telnet 由 RFC 854 定义，它现在是一个用于远程登录的流行应用层协议。它运行在 TCP 之上，被设计成可在任意一对主机之间工作。Telnet 与我们第 2 章讨论的批量数据传输应用不同，它是一个交互式应用。我们在此讨论一个 Telnet 例子，因为该例子很好地阐述 TCP 的序号与确认号。我们注意到许多用户现在更愿意采用 SSH 协议而不是 Telnet，因为在 Telnet 连接中发送的数据（包括口令!）是没有加密的，使得 Telnet 易于受到窃听攻击（如在 8.7 节中讨论的那样)。

假设主机 A 发起一个与主机 B 的 Telnet 会话。因为是主机 A 发起该会话，因此它被标记为客户，而主机 B 被标记为服务器。（在客户端的）用户键入的每个字符都会被发送至远程主机；远程主机将回送每个字符的副本给客户，并将这些字符显示在 Telnet 用户的屏幕上。这种“回显”（echo back）用于确保由 Telnet 用户发送的字符已经被远程主机收到并在远程站点上得到处理。因此，在从用户击键到字符被显示在用户屏幕上这段时间内，每个字符在网络中传输了两次。

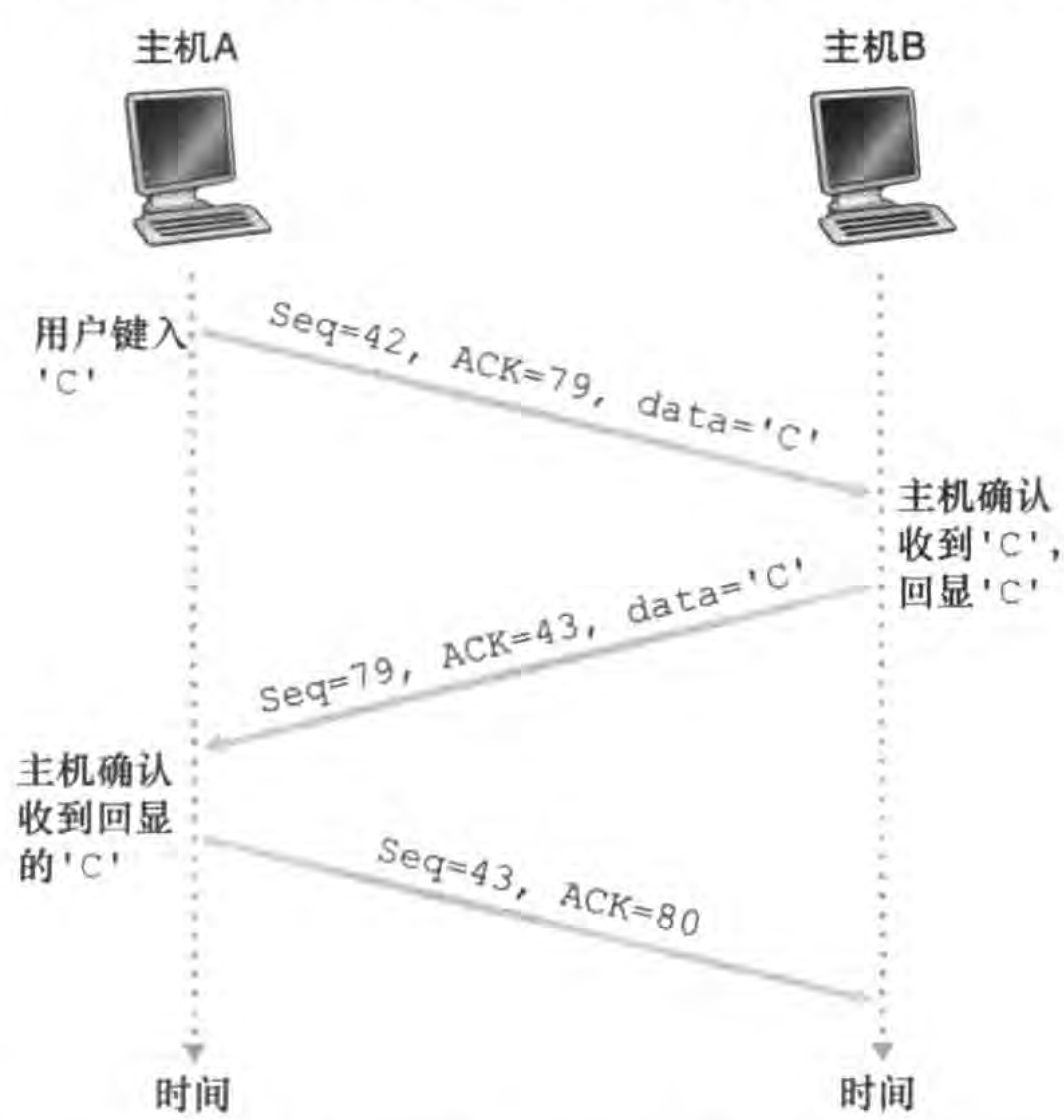


图 3-31　一个经 TCP 的简单 Telnet 应用的确认号和序号

现在假设用户输入了一个字符 'C'，然后喝起了咖啡。我们考察一下在客户与服务器之间发送的 TCP 报文段。如图 3-31

所示，假设客户和服务器的起始序号分别是42和79。前面讲过，一个报文段的序号就是该报文段数据字段首字节的序号。因此，客户发送的第一个报文段的序号为42，服务器发送的第一个报文段的序号为79。前面讲过，确认号就是主机正在等待的数据的下一个字节序号。在TCP连接建立后但没有发送任何数据之前，该客户等待字节79，而该服务器等待字节42。

如图3-31中所示，共发送3个报文段。第一个报文段是由客户发往服务器，在它的数据字段里包含一字节的字符'C'的ASCII码。如我们刚讲到的那样，第一个报文段的序号字段里是42。另外，由于客户还没有接收到来自服务器的任何数据，因此该第一个报文段中的确认号字段中是79。

第二个报文段是由服务器发往客户。它有两个目的：首先它是为该服务器所收到数据提供一个确认。通过在确认号字段中填入43，服务器告诉客户它已经成功地收到字节42及以前的所有字节，现在正等待着字节43的出现。该报文段的第二个目的是回显字符'C'。因此，在第二个报文段的数据字段里填入的是字符'C'的ASCII码。第二个报文段的序号为79，它是该TCP连接上从服务器到客户的数据流的起始序号，这也正是服务器要发送的第一个字节的数据。值得注意的是，对客户到服务器的数据的确认被装载在一个承载服务器到客户的数据的报文段中；这种确认被称为是被**捎带**（piggybacked）在服务器到客户的数据报文段中的。

第三个报文段是从客户发往服务器的。它的唯一目的是确认已从服务器收到的数据。（前面讲过，第二个报文段中包含的数据是字符'C'，是从服务器到客户的。）该报文段的数据字段为空（即确认信息没有被任何从客户到服务器的数据所捎带）。该报文段的确认号字段填入的是80，因为客户已经收到了字节流中序号为79及以前的字节，它现在正等待着字节80的出现。你可能认为这有点奇怪，即使该报文段里没有数据还仍有序号。这是因为TCP存在序号字段，报文段需要填入某个序号。

3.5.3 往返时间的估计与超时

TCP如同前面3.4节所讲的rdt协议一样，它采用超时/重传机制来处理报文段的丢失问题。尽管这在概念上简单，但是当在如TCP这样的实际协议中实现超时/重传机制时还是会产生许多微妙的问题。也许最明显的一个问题就是超时间隔长度的设置。显然，超时间隔必须大于该连接的往返时间（RTT），即从一个报文段发出到它被确认的时间。否则会造成不必要的重传。但是这个时间间隔到底应该是多大呢？刚开始时应如何估计往返时间呢？是否应该为所有未确认的报文段各设一个定时器？问题竟然如此之多！我们在本节中的讨论基于[Jacobson 1988]中有关TCP的工作以及IETF关于管理TCP定时器的建议[RFC 6298]。

1. 估计往返时间

我们开始学习TCP定时器的管理问题，要考虑一下TCP是如何估计发送方与接收方之间往返时间的。这是通过如下方法完成的。报文段的样本RTT（表示为SampleRTT）就是从某报文段被发出（即交给IP）到对该报文段的确认被收到之间的时间量。大多数TCP的实现仅在某个时刻做一次SampleRTT测量，而不是为每个发送的报文段测量一个SampleRTT。这就是说，在任意时刻，仅为一个已发送的但目前尚未被确认的报文段估计SampleRTT，从而产生一个接近每个RTT的新SampleRTT值。另外，TCP决不为已被重传的报文段计算SampleRTT；它仅为传输一次的报文段测量SampleRTT[Kan 1987]。（本章后面

的一个习题请你考虑一下为什么要这么做。)

显然，由于路由器的拥塞和端系统负载的变化，这些报文段的 SampleRTT 值会随之波动。由于这种波动，任何给定的 SampleRTT 值也许都是非典型的。因此，为了估计一个典型的 RTT，自然要采取某种对 SampleRTT 取平均的办法。TCP 维持一个 SampleRTT 均值(称为 EstimatedRTT)。一旦获得一个新 SampleRTT 时，TCP 就会根据下列公式来更新 EstimatedRTT：

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

上面的公式是以编程语言的语句方式给出的，即 EstimatedRTT 的新值是由以前的 EstimatedRTT 值与 SampleRTT 新值加权组合而成的。在 [RFC 6298] 中给出的 α 推荐值是 $\alpha = 0.125$ (即 1/8)，这时上面的公式变为：

$$\text{EstimatedRTT} = 0.875 \cdot \text{EstimatedRTT} + 0.125 \cdot \text{SampleRTT}$$

值得注意的是，EstimatedRTT 是一个 SampleRTT 值的加权平均值。如在本章后面习题中讨论的那样，这个加权平均对最近的样本赋予的权值要大于对旧样本赋予的权值。这是很自然的，因为越近的样本越能更好地反映网络的当前拥塞情况。从统计学观点讲，这种平均被称为**指数加权移动平均** (Exponential Weighted Moving Average，EWMA)。在 EWMA 中的“指数”一词看起来是指一个给定的 SampleRTT 的权值在更新的过程中呈指数型快速衰减。在课后习题中，将要求你推导出 EstimatedRTT 的指数表达形式。

图 3-32 显示了当 $\alpha = 1/8$ 时，在 gaia. cs. umass. edu (在美国马萨诸塞州的 Amherst) 与 fantasia. eurecom. fr (在法国南部) 之间的一条 TCP 连接上的 SampleRTT 值与 EstimatedRTT 值。显然，SampleRTT 的变化在 EstimatedRTT 的计算中趋于平缓了。

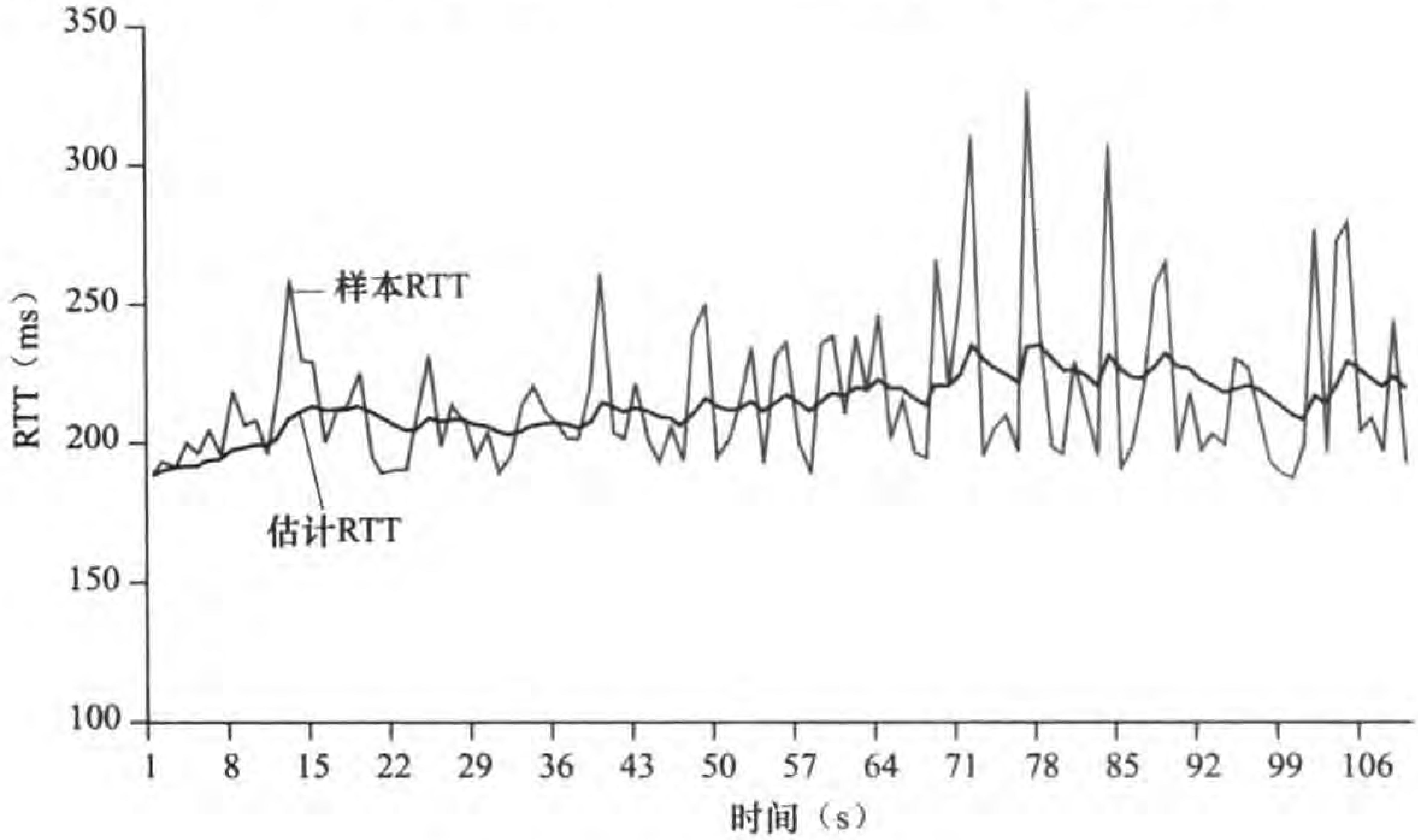


图 3-32　RTT 样本和 RTT 估计

除了估算 RTT 外，测量 RTT 的变化也是有价值的。[RFC 6298] 定义了 RTT 偏差 DevRTT，用于估算 SampleRTT 一般会偏离 EstimatedRTT 的程度：

$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}|$$

注意到 DevRTT 是一个 SampleRTT 与 EstimatedRTT 之间差值的 EWMA。如果 SampleRTT 值波动较小，那么 DevRTT 的值就会很小；另一方面，如果波动很大，那么 DevRTT 的值就会很大。β 的推荐值为 0.25。

2. 设置和管理重传超时间隔

假设已经给出了 EstimatedRTT 值和 DevRTT 值，那么 TCP 超时间隔应该用什么值呢？很明显，超时间隔应该大于等于 EstimatedRTT，否则，将造成不必要的重传。但是超时间隔也不应该比 EstimatedRTT 大太多，否则当报文段丢失时，TCP 不能很快地重传该报文段，导致数据传输时延大。因此要求将超时间隔设为 EstimatedRTT 加上一定余量。当 SampleRTT 值波动较大时，这个余量应该大些；当波动较小时，这个余量应该小些。因此，DevRTT 值应该在这里发挥作用了。在 TCP 的确定重传超时间隔的方法中，所有这些因素都考虑到了：

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$

推荐的初始 TimeoutInterval 值为 1 秒［RFC 6298］。同时，当出现超时后，TimeoutInterval 值将加倍，以免即将被确认的后继报文段过早出现超时。然而，只要收到报文段并更新 EstimatedRTT，就使用上述公式再次计算 TimeoutInterval。

实践原则

与我们在 3.4 节中所学的方法很像，TCP 通过使用肯定确认与定时器来提供可靠数据传输。TCP 确认正确接收到的数据，而当认为报文段或其确认报文丢失或受损时，TCP 会重传这些报文段。有些版本的 TCP 还有一个隐式 NAK 机制（在 TCP 的快速重传机制下，收到对一个特定报文段的 3 个冗余 ACK 就可作为对后面报文段的一个隐式 NAK，从而在超时之前触发对该报文段的重传）。TCP 使用序号以使接收方能识别丢失或重复的报文段。像可靠数据传输协议 rdt3.0 的情况一样，TCP 自己也无法明确地分辨一个报文段或其 ACK 是丢失了还是受损了，或是时延过长了。在发送方，TCP 的响应是相同的：重传有疑问的报文段。

TCP 也使用流水线，使得发送方在任意时刻都可以有多个已发出但还未被确认的报文段存在。我们在前面已经看到，当报文段长度与往返时延之比很小时，流水线可显著地增加一个会话的吞吐量。一个发送方能够具有的未被确认报文段的具体数量是由 TCP 的流量控制和拥塞控制机制决定的。TCP 流量控制将在本节后面讨论；TCP 拥塞控制将在 3.7 节中讨论。此时我们只需知道 TCP 发送方使用了流水线。

3.5.4 可靠数据传输

前面讲过，因特网的网络层服务（IP 服务）是不可靠的。IP 不保证数据报的交付，不保证数据报的按序交付，也不保证数据报中数据的完整性。对于 IP 服务，数据报能够溢出路由器缓存而永远不能到达目的地，数据报也可能是乱序到达，而且数据报中的比特可能损坏（由 0 变为 1 或者相反）。由于运输层报文段是被 IP 数据报携带着在网络中传输的，所以运输层的报文段也会遇到这些问题。

TCP 在 IP 不可靠的尽力而为服务之上创建了一种**可靠数据传输服务**（reliable data transfer service）。TCP 的可靠数据传输服务确保一个进程从其接收缓存中读出的数据流是无损坏、无间隙、非冗余和按序的数据流；即该字节流与连接的另一方端系统发送出的字节流是完全相同。TCP 提供可靠数据传输的方法涉及我们在 3.4 节中所学的许多原理。

在我们前面研发可靠数据传输技术时，曾假定每一个已发送但未被确认的报文段都与一

个定时器相关联，这在概念上是最简单的。虽然这在理论上很好，但定时器的管理却需要相当大的开销。因此，推荐的定时器管理过程［RFC 6298］仅使用单一的重传定时器，即使有多个已发送但还未被确认的报文段。在本节中描述的 TCP 协议遵循了这种单一定时器的推荐。

我们将以两个递增的步骤来讨论 TCP 是如何提供可靠数据传输的。我们先给出一个 TCP 发送方的高度简化的描述，该发送方只用超时来恢复报文段的丢失；然后再给出一个更全面的描述，该描述中除了使用超时机制外，还使用冗余确认技术。在接下来的讨论中，我们假定数据仅向一个方向发送，即从主机 A 到主机 B，且主机 A 在发送一个大文件。

图 3-33 给出了一个 TCP 发送方高度简化的描述。我们看到在 TCP 发送方有 3 个与发送和重传有关的主要事件：从上层应用程序接收数据；定时器超时和收到 ACK。一旦第一个主要事件发生，TCP 从应用程序接收数据，将数据封装在一个报文段中，并把该报文段交给 IP。注意到每一个报文段都包含一个序号，如 3.5.2 节所讲的那样，这个序号就是该报文段第一个数据字节的字节流编号。还要注意到如果定时器还没有为某些其他报文段而运行，则当报文段被传给 IP 时，TCP 就启动该定时器。（将定时器想象为与最早的未被确认的报文段相关联是有帮助的。）该定时器的过期间隔是 TimeoutInterval，它是由 3.5.3 节中所描述的 EstimatedRTT 和 DevRTT 计算得出的。

```
/* 假设发送方不受TCP流量和拥塞控制的限制，来自上层数据的长度小于MSS，且数据传送只在一个
方向进行。*/

NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop  (永远)  {
    switch (事件)

        事件：从上面应用程序接收到数据e
            生成具有序号NextSeqNum的TCP报文段
            if (定时器当前没有运行)
                启动定时器
            向IP传递报文段
            NextSeqNum=NextSeqNum+length(data)
            break;

        事件：定时器超时
            重传具有最小序号但仍未应答的报文段
            启动定时器
            break;

        事件：收到ACK，具有ACK字段值y
            if (y > SendBase) {
                SendBase=y
                if (当前仍无任何应答报文段)
                    启动定时器
                }
            break;

    } /* 结束永远循环 */
```

图 3-33　简化的 TCP 发送方

第二个主要事件是超时。TCP 通过重传引起超时的报文段来响应超时事件。然后 TCP 重启定时器。

TCP 发送方必须处理的第三个主要事件是，到达一个来自接收方的确认报文段（ACK）（更确切地说，是一个包含了有效 ACK 字段值的报文段）。当该事件发生时，TCP 将 ACK 的值 y 与它的变量 SendBase 进行比较。TCP 状态变量 SendBase 是最早未被确认的字节的序号。（因此 SendBase - 1 是指接收方已正确按序接收到的数据的最后一个字节的序号。）如前面指出的那样，TCP 采用累积确认，所以 y 确认了字节编号在 y 之前的所有字节都已经收到。如果 y > SendBase，则该 ACK 是在确认一个或多个先前未被确认的报文段。因此发送方更新它的 SendBase 变量；如果当前有未被确认的报文段，TCP 还要重新启动定时器。

1. 一些有趣的情况

我们刚刚描述了一个关于 TCP 如何提供可靠数据传输的高度简化的版本。但即使这种高度简化的版本，仍然存在着许多微妙之处。为了较好地感受该协议的工作过程，我们来看几种简单情况。图 3-34 描述了第一种情况，主机 A 向主机 B 发送一个报文段。假设该报文段的序号是 92，而且包含 8 字节数据。在发出该报文段之后，主机 A 等待一个来自主机 B 的确认号为 100 的报文段。虽然 A 发出的报文段在主机 B 上被收到，但从主机 B 发往主机 A 的确认报文丢失了。在这种情况下，超时事件就会发生，主机 A 会重传相同的报文段。当然，当主机 B 收到该重传的报文段时，它将通过序号发现该报文段包含了早已收到的数据。因此，主机 B 中的 TCP 将丢弃该重传的报文段中的这些字节。

在第二种情况中，如图 3-35 所示，主机 A 连续发回了两个报文段。第一个报文段序号是 92，包含 8 字节数据；第二个报文段序号是 100，包含 20 字节数据。假设两个报文段都完好无损地到达主机 B，并且主机 B 为每一个报文段分别发送一个确认。第一个确认报文的确认号是 100，第二个确认报文的确认号是 120。现在假设在超时之前这两个报文段中没有一个确认报文到达主机 A。当超时事件发生时，主机 A 重传序号 92 的第一个报文段，并重启定时器。只要第二个报文段的 ACK 在新的超时发生以前到达，则第二个报文段将不会被重传。

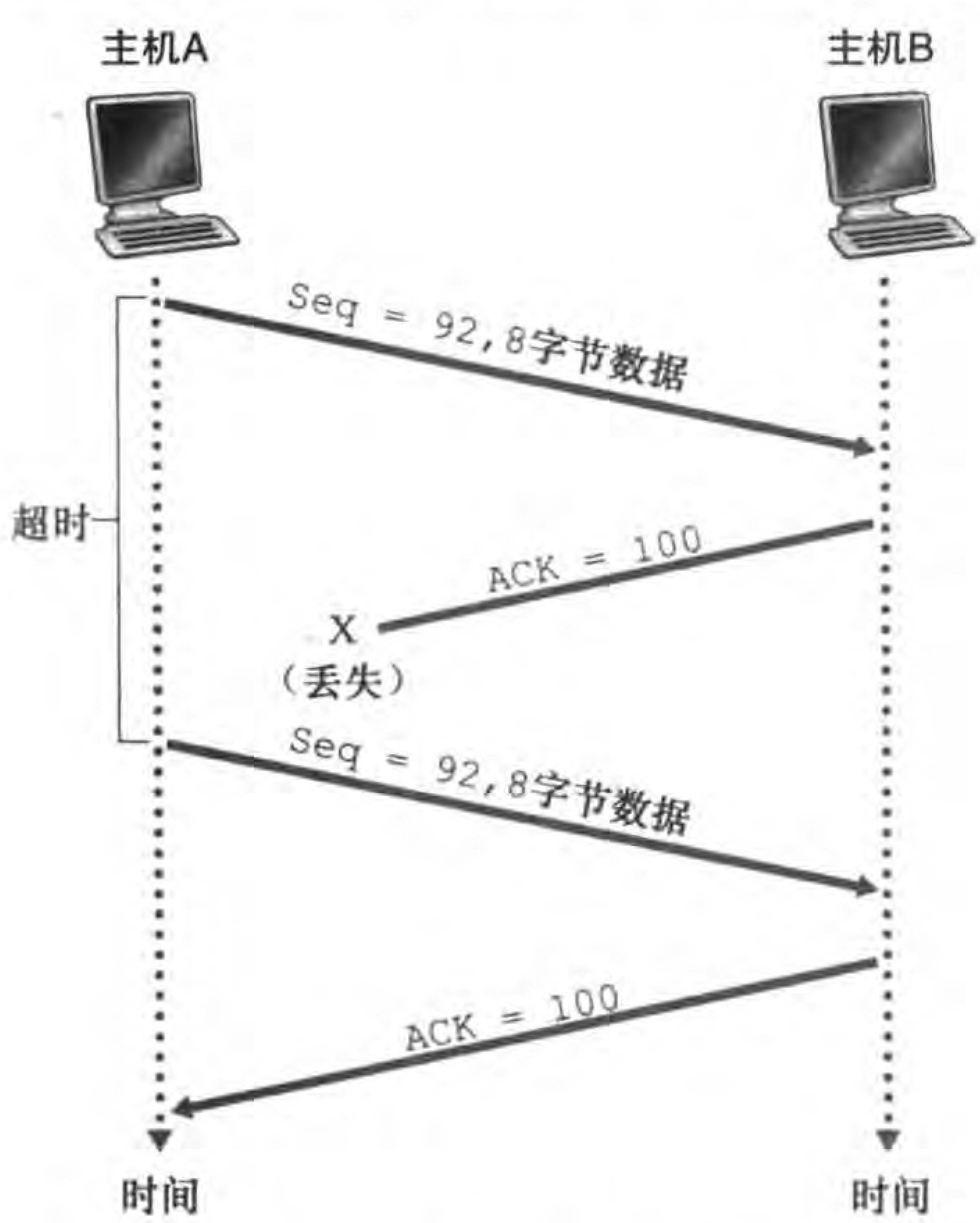


图 3-34 由于确认丢失而重传

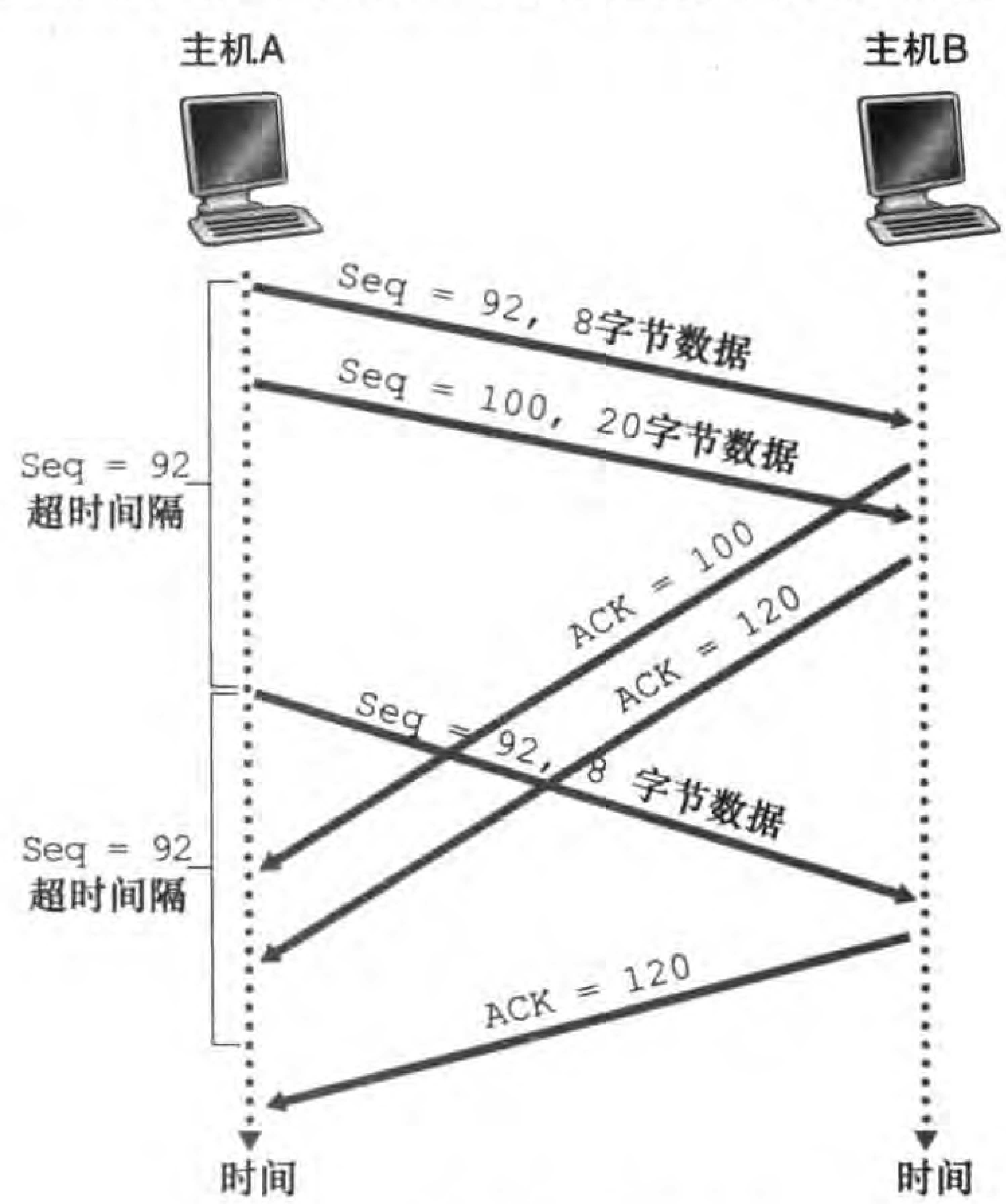


图 3-35 报文段 100 没有重传

在第三种也是最后一种情况中，假设主机 A 与在第二种情况中完全一样，发送两个报

文段。第一个报文段的确认报文在网络丢失，但在超时事件发生之前主机A收到一个确认号为120的确认报文。主机A因而知道主机B已经收到了序号为119及之前的所有字节；所以主机A不会重传这两个报文段中的任何一个。这种情况在图3-36中进行了图示。

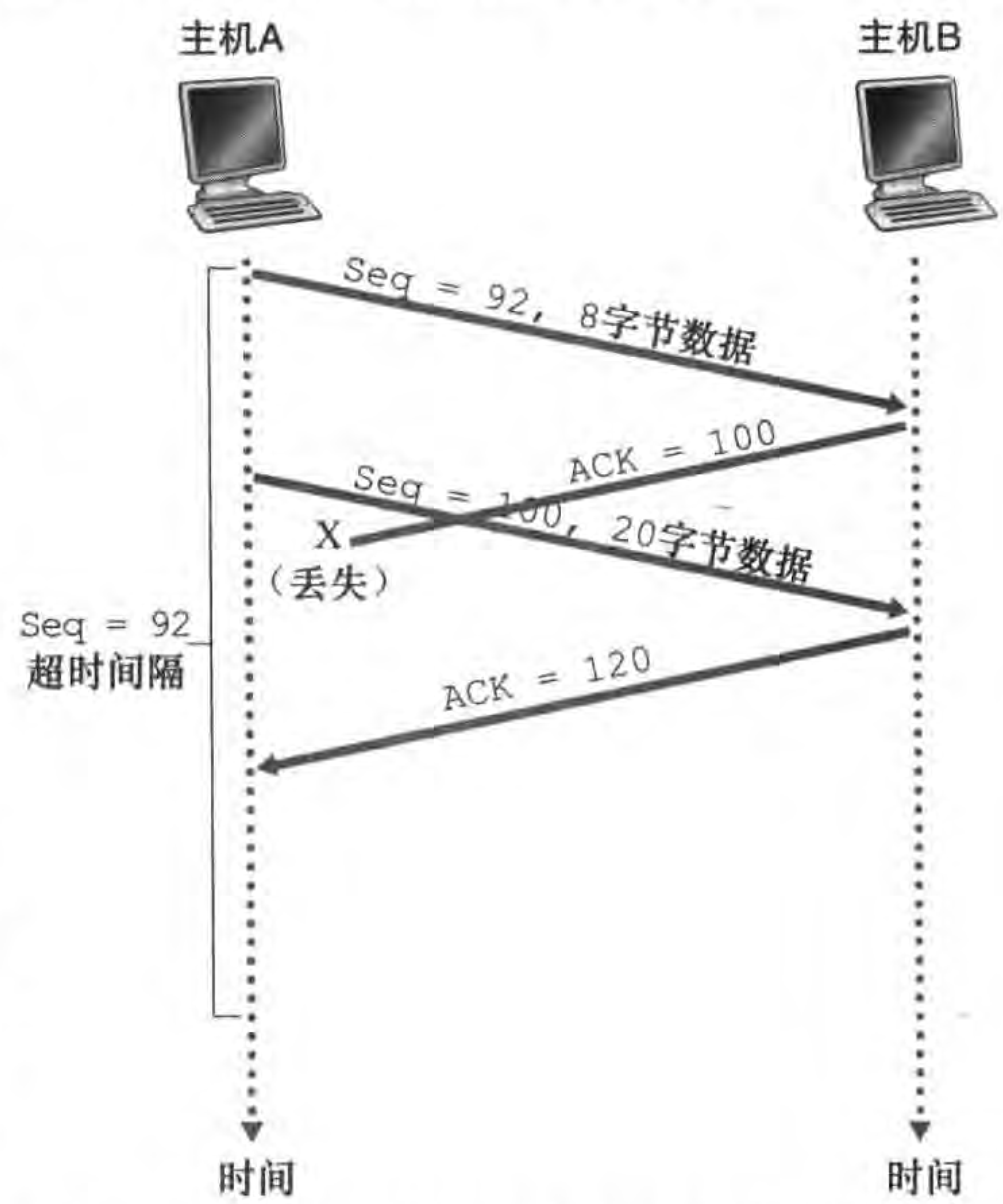


图3-36　累积确认避免了第一个报文段的重传

2. 超时间隔加倍

我们现在讨论一下在大多数TCP实现中所做的一些修改。首先关注的是在定时器时限过期后超时间隔的长度。在这种修改中，每当超时事件发生时，如前所述，TCP重传具有最小序号的还未被确认的报文段。只是每次TCP重传时都会将下一次的超时间隔设为先前值的两倍，而不是用从EstimatedRTT和DevRTT推算出的值（如在3.5.3节中所描述的）。例如，假设当定时器第一次过期时，与最早的未被确认的报文段相关联的TimeoutInterval是0.75秒。TCP就会重传该报文段，并把新的过期时间设置为1.5秒。如果1.5秒后定时器又过期了，则TCP将再次重传该报文段，并把过期时间设置为3.0秒。因此，超时间隔在每次重传后会呈指数型增长。然而，每当定时器在另两个事件（即收到上层应用的数据和收到ACK）中的任意一个启动时，TimeoutInterval由最近的EstimatedRTT值与DevRTT值推算得到。

这种修改提供了一个形式受限的拥塞控制。（更复杂的TCP拥塞控制形式将在3.7节中学习。）定时器过期很可能是由网络拥塞引起的，即太多的分组到达源与目的地之间路径上的一台（或多台）路由器的队列中，造成分组丢失或长时间的排队时延。在拥塞的时候，如果源持续重传分组，会使拥塞更加严重。相反，TCP使用更文雅的方式，每个发送方的重传都是经过越来越长的时间间隔后进行的。当我们在第6章学习CSMA/CD时，将看到以太网采用了类似的思路。

3. 快速重传

超时触发重传存在的问题之一是超时周期可能相对较长。当一个报文段丢失时，这种长超时周期迫使发送方延迟重传丢失的分组，因而增加了端到端时延。幸运的是，发送方通常可在超时事件发生之前通过注意所谓冗余ACK来较好地检测到丢包情况。**冗余ACK**（duplicate ACK）就是再次确认某个报文段的ACK，而发送方先前已经收到对该报文段的确认。要理解发送方对冗余ACK的响应，我们必须首先看一下接收方为什么会发送冗余ACK。表3-2总结了TCP接收方的ACK生成策略［RFC 5681］。当TCP接收方收到一个具有这样序号的报文段时，即其序号大于下一个所期望的、按序的报文段，它检测到了数据流中的一个间隔，这就是说有报文段丢失。这个间隔可能是由于在网络中报文段丢失或重新排序造成的。因为TCP不使用否定确认，所以接收方不能向发送方发回一个显式的否定确认。相反，它只是对已经接收到的最后一个按序字节数据进行重复确认（即产生一个冗余ACK）即可。（注意到在

表 3-2 中允许接收方不丢弃失序报文段。)

表 3-2 产生 TCP ACK 的建议 [RFC 5681]

事件	TCP 接收方动作
具有所期望序号的按序报文段到达。所有在期望序号及以前的数据都已经被确认	延迟的 ACK。对另一个按序报文段的到达最多等待 500ms。如果下一个按序报文段在这个时间间隔内没有到达，则发送一个 ACK
具有所期望序号的按序报文段到达。另一个按序报文段等待 ACK 传输	立即发送单个累积 ACK，以确认两个按序报文段
比期望序号大的失序报文段到达。检测出间隔	立即发送冗余 ACK，指示下一个期待字节的序号（其为间隔的低端的序号）
能部分或完全填充接收数据间隔的报文段到达	倘若该报文段起始于间隔的低端，则立即发送 ACK

因为发送方经常一个接一个地发送大量的报文段，如果一个报文段丢失，就很可能引起许多一个接一个的冗余 ACK。如果 TCP 发送方接收到对相同数据的 3 个冗余 ACK，它把这当作一种指示，说明跟在这个已被确认过 3 次的报文段之后的报文段已经丢失。(在课后习题中，我们将考虑为什么发送方等待 3 个冗余 ACK，而不是仅仅等待一个冗余 ACK。）一旦收到 3 个冗余 ACK，TCP 就执行**快速重传**（fast retransmit）[RFC 5681]，即在该报文段的定时器过期之前重传丢失的报文段。对于采用快速重传的 TCP，可用下列代码片段代替图 3-33 中的 ACK 收到事件：

```
事件: 收到ACK,具有ACK字段值y
    if (y > SendBase) {
    SendBase=y
    if (当前仍无任何应答报文段)
        启动定时器
        }
    else {/*对已经确认的报文段的一个冗余ACK */
        对y收到的冗余ACK数加1
        if (对y==3收到的冗余ACK数)
            /* TCP快速重传 */
            重新发送具有序号y的报文段
        }
   break;
```

前面讲过，当在如 TCP 这样一个实际协议中实现超时/重传机制时，会产生许多微妙的问题。上面的过程是在超过 20 年的 TCP 定时器使用经验的基础上演化而来的，读者应当理解实际情况确实是这样的。

4. 是回退 *N* 步还是选择重传

考虑下面这个问题来结束有关 TCP 差错恢复机制的学习：TCP 是一个 GBN 协议还是一个 SR 协议？前面讲过，TCP 确认是累积式的，正确接收但失序的报文段是不会被接收方逐个确认的。因此，如图 3-33 所示（也可参见图 3-19)，TCP 发送方仅需维持已发送过但未被确认的字节的最小序号（SendBase）和下一个要发送的字节的序号（NextSeqNum)。在这种意义下，TCP 看起来更像一个 GBN 风格的协议。但是 TCP 和 GBN 协议之间有着一些显著的区别。许多 TCP 实现会将正确接收但失序的报文段缓存起来 [Stevens 1994]。另外考虑一下，当发送方发送的一组报文段 1，2，…，*N*，并且所有的

报文段都按序无差错地到达接收方时会发生的情况。进一步假设对分组 $n<N$ 的确认报文丢失，但是其余 $N-1$ 个确认报文在分别超时以前到达发送端，这时又会发生的情况。在该例中，GBN 不仅会重传分组 n，还会重传所有后继的分组 $n+1$，$n+2$，…，N。在另一方面，TCP 将重传至多一个报文段，即报文段 n。此外，如果对报文段 $n+1$ 的确认报文在报文段 n 超时之前到达，TCP 甚至不会重传报文段 n。

对 TCP 提出的一种修改意见是所谓的**选择确认**（selective acknowledgment）[RFC 2018]，它允许 TCP 接收方有选择地确认失序报文段，而不是累积地确认最后一个正确接收的有序报文段。当将该机制与选择重传机制结合起来使用时（即跳过重传那些已被接收方选择性地确认过的报文段），TCP 看起来就很像我们通常的 SR 协议。因此，TCP 的差错恢复机制也许最好被分类为 GBN 协议与 SR 协议的混合体。

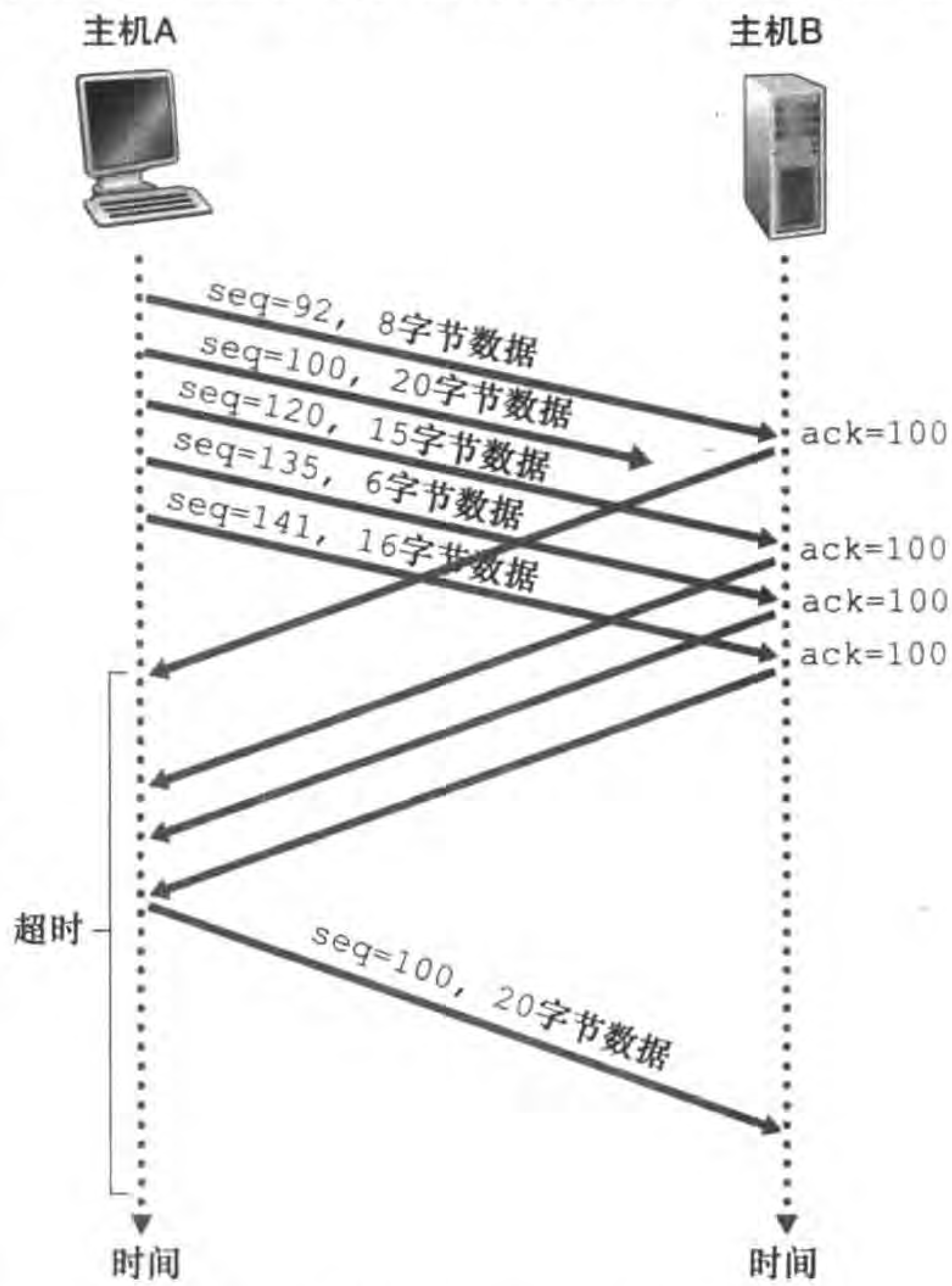


图 3-37　快速重传：在某报文段的定时器过期之前重传丢失的报文段

3.5.5　流量控制

前面讲过，一条 TCP 连接的每一侧主机都为该连接设置了接收缓存。当该 TCP 连接收到正确、按序的字节后，它就将数据放入接收缓存。相关联的应用进程会从该缓存中读取数据，但不必是数据刚一到达就立即读取。事实上，接收方应用也许正忙于其他任务，甚至要过很长时间后才去读取该数据。如果某应用程序读取数据时相对缓慢，而发送方发送得太多、太快，发送的数据就会很容易地使该连接的接收缓存溢出。

TCP 为它的应用程序提供了**流量控制服务**（flow-control service）以消除发送方使接收方缓存溢出的可能性。流量控制因此是一个速度匹配服务，即发送方的发送速率与接收方应用程序的读取速率相匹配。前面提到过，TCP 发送方也可能因为 IP 网络的拥塞而被遏制；这种形式的发送方的控制被称为**拥塞控制**（congestion control），我们将在 3.6 节和 3.7 节详细地讨论这个主题。即使流量控制和拥塞控制采取的动作非常相似（对发送方的遏制），但是它们显然是针对完全不同的原因而采取的措施。不幸的是，许多作者把这两个术语混用，理解力强的读者会明智地区分这两种情况。现在我们来讨论 TCP 如何提供流量控制服务的。为了能从整体上看问题，我们在本节都假设 TCP 是这样实现的，即 TCP 接收方丢弃失序的报文段。

TCP 通过让发送方维护一个称为**接收窗口**（receive window）的变量来提供流量控制。通俗地说，接收窗口用于给发送方一个指示——该接收方还有多少可用的缓存空间。因为 TCP 是全双工通信，在连接两端的发送方都各自维护一个接收窗口。我们在文件传输的情况下研究接收窗口。假设主机 A 通过一条 TCP 连接向主机 B 发送一个大文件。主机 B 为该连接分配了一个接收缓存，并用 RcvBuffer 来表示其大小。主机 B 上的应用进程不时地

从该缓存中读取数据。我们定义以下变量：

- LastByteRead：主机 B 上的应用进程从缓存读出的数据流的最后一个字节的编号。
- LastByteRcvd：从网络中到达的并且已放入主机 B 接收缓存中的数据流的最后一个字节的编号。

由于 TCP 不允许已分配的缓存溢出，下式必须成立：

$$\text{LastByteRcvd} - \text{LastByteRead} \leqslant \text{RcvBuffer}$$

接收窗口用 rwnd 表示，根据缓存可用空间的数量来设置：

$$\text{rwnd} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$$

由于该空间是随着时间变化的，所以 rwnd 是动态的。图 3-38 对变量 rwnd 进行了图示。

连接是如何使用变量 rwnd 来提供流量控制服务的呢？主机 B 通过把当前的 rwnd 值放入它发给主机 A 的报文段接收窗口字段中，通知主机 A 它在该连接的缓存中还有多少可用空间。开始时，主机 B 设定 rwnd = RcvBuffer。注意到为了实现这一点，主机 B 必须跟踪几个与连接有关的变量。

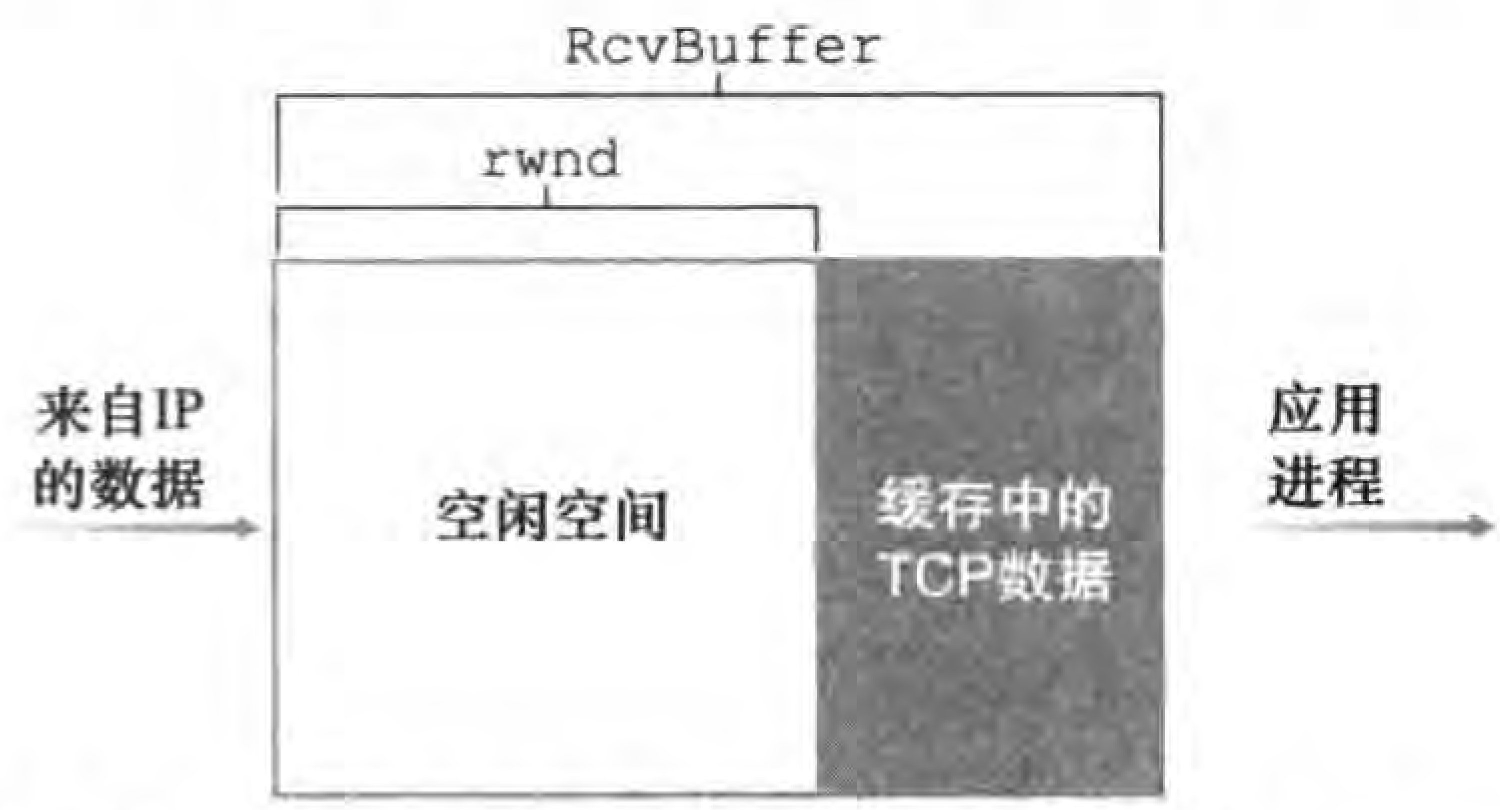


图 3-38 接收窗口（rwnd）和接收缓存（RcvBuffer）

主机 A 轮流跟踪两个变量，LastByteSent 和 LastByteAcked，这两个变量的意义很明显。注意到这两个变量之间的差 LastByteSent - LastByteAcked，就是主机 A 发送到连接中但未被确认的数据量。通过将未确认的数据量控制在值 rwnd 以内，就可以保证主机 A 不会使主机 B 的接收缓存溢出。因此，主机 A 在该连接的整个生命周期须保证：

$$\text{LastByteSent} - \text{LastByteAcked} \leqslant \text{rwnd}$$

对于这个方案还存在一个小小的技术问题。为了理解这一点，假设主机 B 的接收缓存已经存满，使得 rwnd = 0。在将 rwnd = 0 通告给主机 A 之后，还要假设主机 B 没有任何数据要发给主机 A。此时，考虑会发生什么情况。因为主机 B 上的应用进程将缓存清空，TCP 并不向主机 A 发送带有 rwnd 新值的新报文段；事实上，TCP 仅当在它有数据或有确认要发时才会发送报文段给主机 A。这样，主机 A 不可能知道主机 B 的接收缓存已经有新的空间了，即主机 A 被阻塞而不能再发送数据！为了解决这个问题，TCP 规范中要求：当主机 B 的接收窗口为 0 时，主机 A 继续发送只有一个字节数据的报文段。这些报文段将会被接收方确认。最终缓存将开始清空，并且确认报文里将包含一个非 0 的 rwnd 值。

位于 http://www.awl.com/kurose-ross 的在线站点为本书提供了一个交互式 Java 小程序，用以说明 TCP 接收窗口的运行情况。

描述了 TCP 的流量控制服务以后，我们在此要简要地提一下 UDP 并不提供流量控制，报文段由于缓存溢出可能在接收方丢失。例如，考虑一下从主机 A 上的一个进程向主机 B 上的一个进程发送一系列 UDP 报文段的情形。对于一个典型的 UDP 实现，UDP 将在一个有限大小的缓存中加上报文段，该缓存在相应套接字（进程的门户）“之前”。进程每次从缓存中读取一个完整的报文段。如果进程从缓存中读取报文段的速度不够快，那么缓存将会溢出，并且将丢失报文段。

3.5.6 TCP 连接管理

在本小节中，我们更为仔细地观察如何建立和拆除一条 TCP 连接。尽管这个主题并不特别令人兴奋，但是它很重要，因为 TCP 连接的建立会显著地增加人们感受到的时延（如在 Web 上冲浪时）。此外，许多常见的网络攻击（包括极为流行的 SYN 洪泛攻击）利用了 TCP 连接管理中的弱点。现在我们观察一下一条 TCP 连接是如何建立的。假设运行在一台主机（客户）上的一个进程想与另一台主机（服务器）上的一个进程建立一条连接。客户应用进程首先通知客户 TCP，它想建立一个与服务器上某个进程之间的连接。客户中的 TCP 会用以下方式与服务器中的 TCP 建立一条 TCP 连接：

- 第一步：客户端的 TCP 首先向服务器端的 TCP 发送一个特殊的 TCP 报文段。该报文段中不包含应用层数据。但是在报文段的首部（参见图 3-29）中的一个标志位（即 SYN 比特）被置为 1。因此，这个特殊报文段被称为 SYN 报文段。另外，客户会随机地选择一个初始序号（client_isn），并将此编号放置于该起始的 TCP SYN 报文段的序号字段中。该报文段会被封装在一个 IP 数据报中，并发送给服务器。为了避免某些安全性攻击，在适当地随机化选择 client_isn 方面有着不少有趣的研究［CERT 2001-09］。
- 第二步：一旦包含 TCP SYN 报文段的 IP 数据报到达服务器主机（假定它的确到达了！），服务器会从该数据报中提取出 TCP SYN 报文段，为该 TCP 连接分配 TCP 缓存和变量，并向该客户 TCP 发送允许连接的报文段。（我们将在第 8 章看到，在完成三次握手的第三步之前分配这些缓存和变量，使得 TCP 易于受到称为 SYN 洪泛的拒绝服务攻击。）这个允许连接的报文段也不包含应用层数据。但是，在报文段的首部却包含 3 个重要的信息。首先，SYN 比特被置为 1。其次，该 TCP 报文段首部的确认号字段被置为 client_isn + 1。最后，服务器选择自己的初始序号（server_isn），并将其放置到 TCP 报文段首部的序号字段中。这个允许连接的报文段实际上表明了："我收到了你发起建立连接的 SYN 分组，该分组带有初始序号 client_isn。我同意建立该连接。我自己的初始序号是 server_isn。"该允许连接的报文段被称为 **SYNACK 报文段**（SYNACK segment）。
- 第三步：在收到 SYNACK 报文段后，客户也要给该连接分配缓存和变量。客户主机则向服务器发送另外一个报文段；这最后一个报文段对服务器的允许连接的报文段进行了确认（该客户通过将值 server_isn + 1 放置到 TCP 报文段首部的确认字段中来完成此项工作）。因为连接已经建立了，所以该 SYN 比特被置为 0。该三次握手的第三个阶段可以在报文段负载中携带客户到服务器的数据。

一旦完成这 3 个步骤，客户和服务器主机就可以相互发送包括数据的报文段了。在以后每一个报文段中，SYN 比特都将被置为 0。注意到为了创建该连接，在两台主机之间发送了 3 个分组，如图 3-39 所示。由于这个原因，这种连接创建过程通常被称为 **3 次握手**（three-way handshake）。TCP 3 次握手的几个方面将在课后习题中讨论（为什么需要初始序号？为什么需要 3 次握手，而不是两次握手？）。注意到这样一件事是很有趣的，一个攀岩者和一个保护者（他位于攀岩者的下面，他的任务是处理好攀岩者的安全绳索）就使用了与 TCP 相同的 3 次握手通信协议，以确保在攀岩者开始攀爬前双方都已经准备好了。

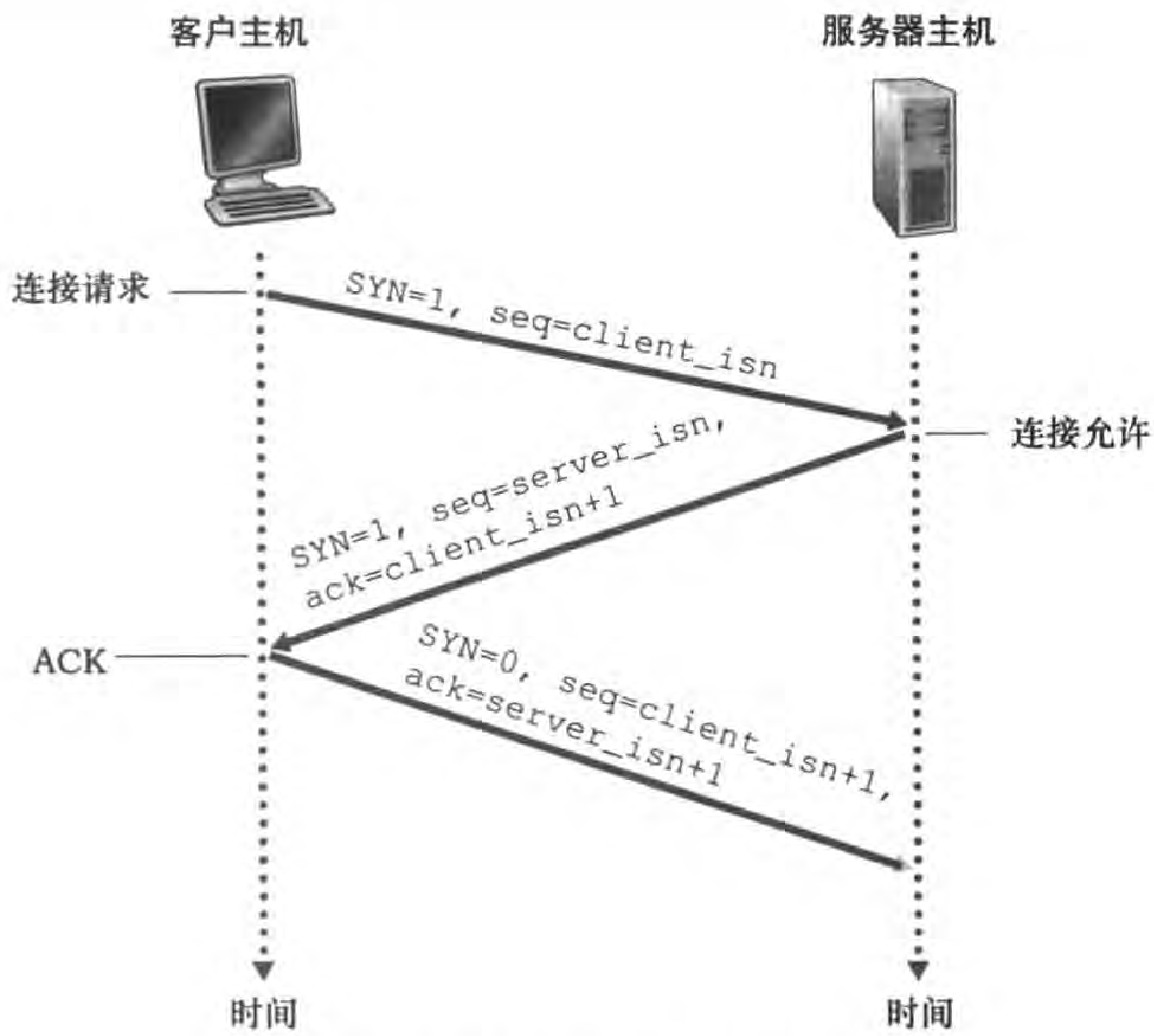


图 3-39 TCP 三次握手：报文段交换

天下没有不散的宴席，对于 TCP 连接也是如此。参与一条 TCP 连接的两个进程中的任何一个都能终止该连接。当连接结束后，主机中的“资源”（即缓存和变量）将被释放。举一个例子，假设某客户打算关闭连接，如图 3-40 所示。客户应用进程发出一个关闭连接命令。这会引起客户 TCP 向服务器进程发送一个特殊的 TCP 报文段。这个特殊的报文段让其首部中的一个标志位即 FIN 比特（参见图 3-29）被设置为 1。当服务器接收到该报文段后，就向发送方回送一个确认报文段。然后，服务器发送它自己的终止报文段，其 FIN 比特被置为 1。最后，该客户对这个服务器的终止报文段进行确认。此时，在两台主机上用于该连接的所有资源都被释放了。

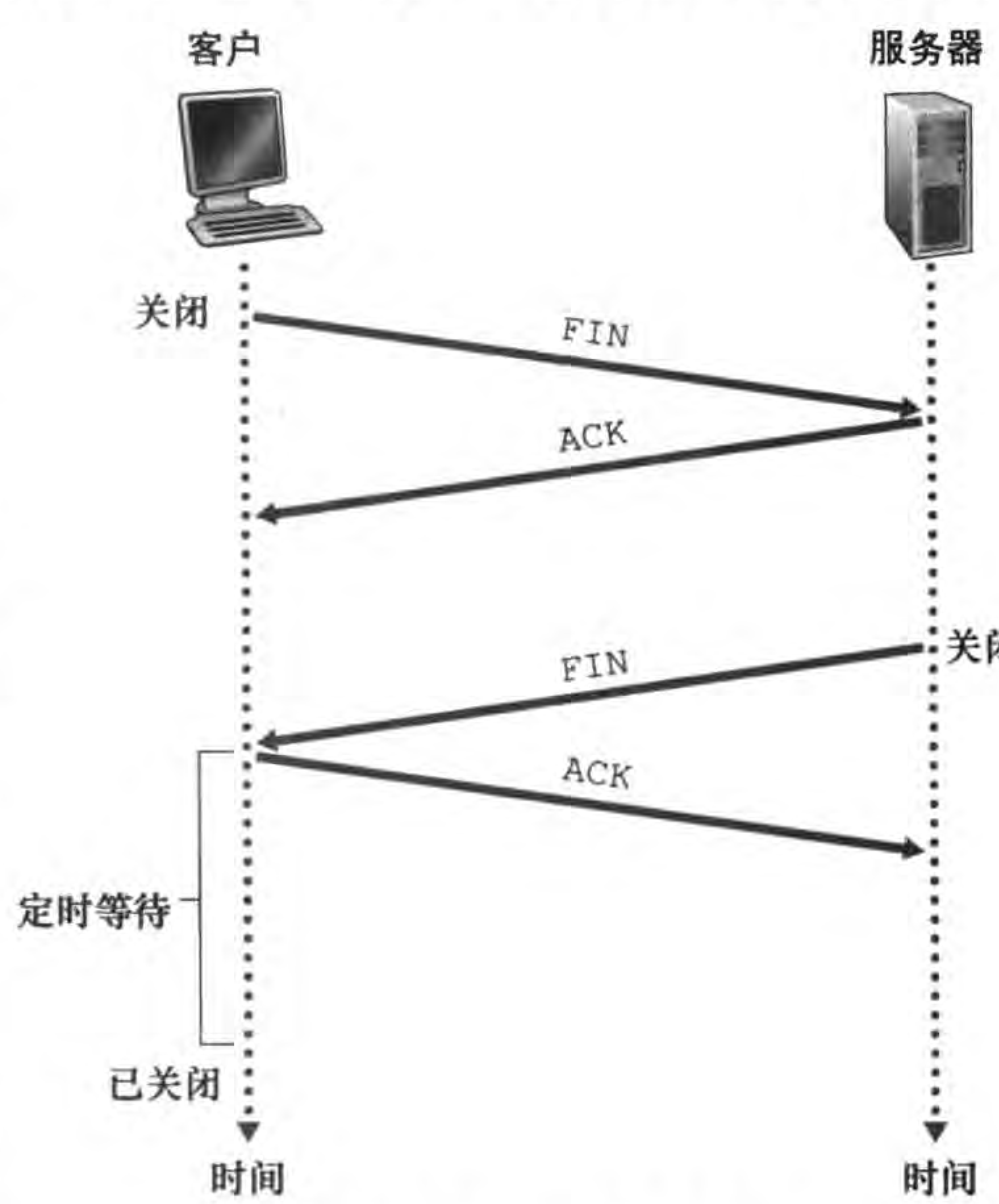


图 3-40 关闭一条 TCP 连接

在一个 TCP 连接的生命周期内，运行在每台主机中的 TCP 协议在各种 **TCP 状态**（TCP state）之间变迁。图 3-41 说明了客户 TCP 会经历的一系列典型 TCP 状态。客户 TCP 开始时处于 CLOSED（关闭）状态。客户的应用程序发起一个新的 TCP 连接（可通过在第 2 章讲过的 Python 例子中创建一个 Socket 对象来完成）。这引起客户中的 TCP 向服务器中的 TCP 发送一个 SYN 报文段。在发送过 SYN 报文段后，客户 TCP 进入了 SYN_SENT 状态。当客户 TCP 处在 SYN_SENT 状态时，它等待来自服务器 TCP 的对客户所发报

文段进行确认且 SYN 比特被置为 1 的一个报文段。收到这样一个报文段之后，客户 TCP 进入 ESTABLISHED（已建立）状态。当处在 ESTABLISHED 状态时，TCP 客户就能发送和接收包含有效载荷数据（即应用层产生的数据）的 TCP 报文段了。

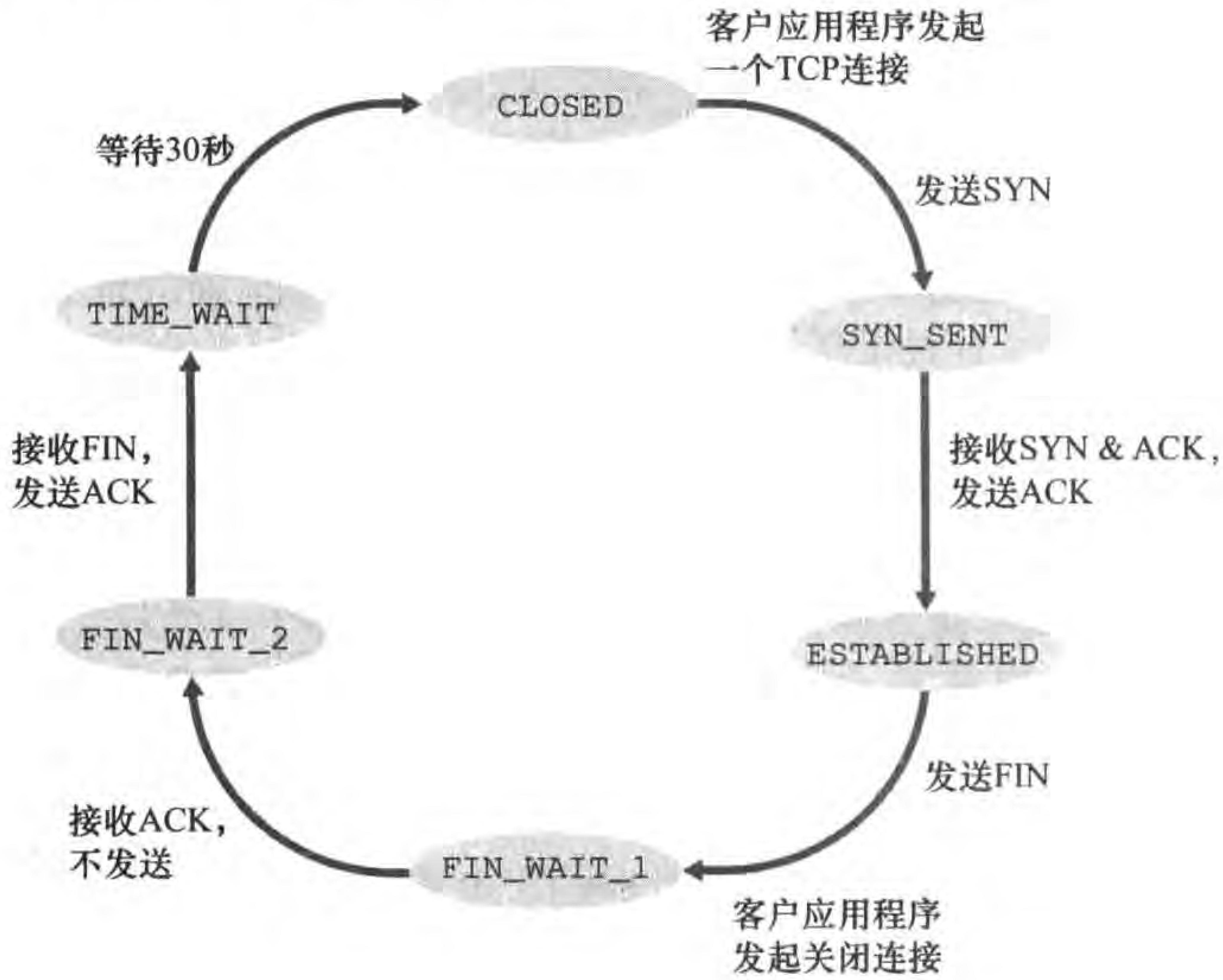


图 3-41 客户 TCP 经历的典型的 TCP 状态序列

假设客户应用程序决定要关闭该连接。（注意到服务器也能选择关闭该连接。）这引起客户 TCP 发送一个带有 FIN 比特被置为 1 的 TCP 报文段，并进入 FIN_WAIT_1 状态。当处在 FIN_WAIT_1 状态时，客户 TCP 等待一个来自服务器的带有确认的 TCP 报文段。当它收到该报文段时，客户 TCP 进入 FIN_WAIT_2 状态。当处在 FIN_WAIT_2 状态时，客户等待来自服务器的 FIN 比特被置为 1 的另一个报文段；当收到该报文段后，客户 TCP 对服务器的报文段进行确认，并进入 TIME_WAIT 状态。假定 ACK 丢失，TIME_WAIT 状态使 TCP 客户重传最后的确认报文。在 TIME_WAIT 状态中所消耗的时间是与具体实现有关的，而典型的值是 30 秒、1 分钟或 2 分钟。经过等待后，连接就正式关闭，客户端所有资源（包括端口号）将被释放。

关注安全性

SYN 洪泛攻击

我们在 TCP 三次握手的讨论中已经看到，服务器为了响应一个收到的 SYN，分配并初始化连接变量和缓存。然后服务器发送一个 SYNACK 进行响应，并等待来自客户的 ACK 报文段。如果某客户不发送 ACK 来完成该三次握手的第三步，最终（通常在一分多钟之后）服务器将终止该半开连接并回收资源。

这种 TCP 连接管理协议为经典的 DoS 攻击即 **SYN 洪泛攻击**（SYN flood attack）提供了环境。在这种攻击中，攻击者发送大量的 TCP SYN 报文段，而不完成第三次握手的步骤。随着这种 SYN 报文段纷至沓来，服务器不断为这些半开连接分配资源（但从未

使用)，导致服务器的连接资源被消耗殆尽。这种 SYN 洪泛攻击是被记载的众多 DoS 攻击中的第一种［CERT SYN 1996］。幸运的是，现在有一种有效的防御系统，称为 **SYN cookie**［RFC 4987］，它们被部署在大多数主流操作系统中。SYN cookie 以下列方式工作：

- 当服务器接收到一个 SYN 报文段时，它并不知道该报文段是来自一个合法的用户，还是一个 SYN 洪泛攻击的一部分。因此服务器不会为该报文段生成一个半开连接。相反，服务器生成一个初始 TCP 序列号，该序列号是 SYN 报文段的源和目的 IP 地址与端口号以及仅有该服务器知道的秘密数的一个复杂函数（散列函数）。这种精心制作的初始序列号被称为“cookie”。服务器则发送具有这种特殊初始序列号的 SYNACK 分组。**重要的是，服务器并不记忆该 cookie 或任何对应于 SYN 的其他状态信息。**
- 如果客户是合法的，则它将返回一个 ACK 报文段。当服务器收到该 ACK，需要验证该 ACK 是与前面发送的某些 SYN 相对应的。如果服务器没有维护有关 SYN 报文段的记忆，这是怎样完成的呢？正如你可能猜测的那样，它是借助于 cookie 来做到的。前面讲过对于一个合法的 ACK，在确认字段中的值等于在 SYNACK 字段（此时为 cookie 值）中的值加 1（参见图 3-39）。服务器则将使用在 SYNACK 报文段中的源和目的地 IP 地址与端口号（它们与初始的 SYN 中的相同）以及秘密数运行相同的散列函数。如果该函数的结果加 1 与在客户的 SYNACK 中的确认（cookie）值相同的话，服务器认为该 ACK 对应于较早的 SYN 报文段，因此它是合法的。服务器则生成一个具有套接字的全开的连接。
- 在另一方面，如果客户没有返回一个 ACK 报文段，则初始的 SYN 并没有对服务器产生危害，因为服务器没有为它分配任何资源。

图 3-42 图示了服务器端的 TCP 通常要经历的一系列状态，其中假设客户开始连接拆

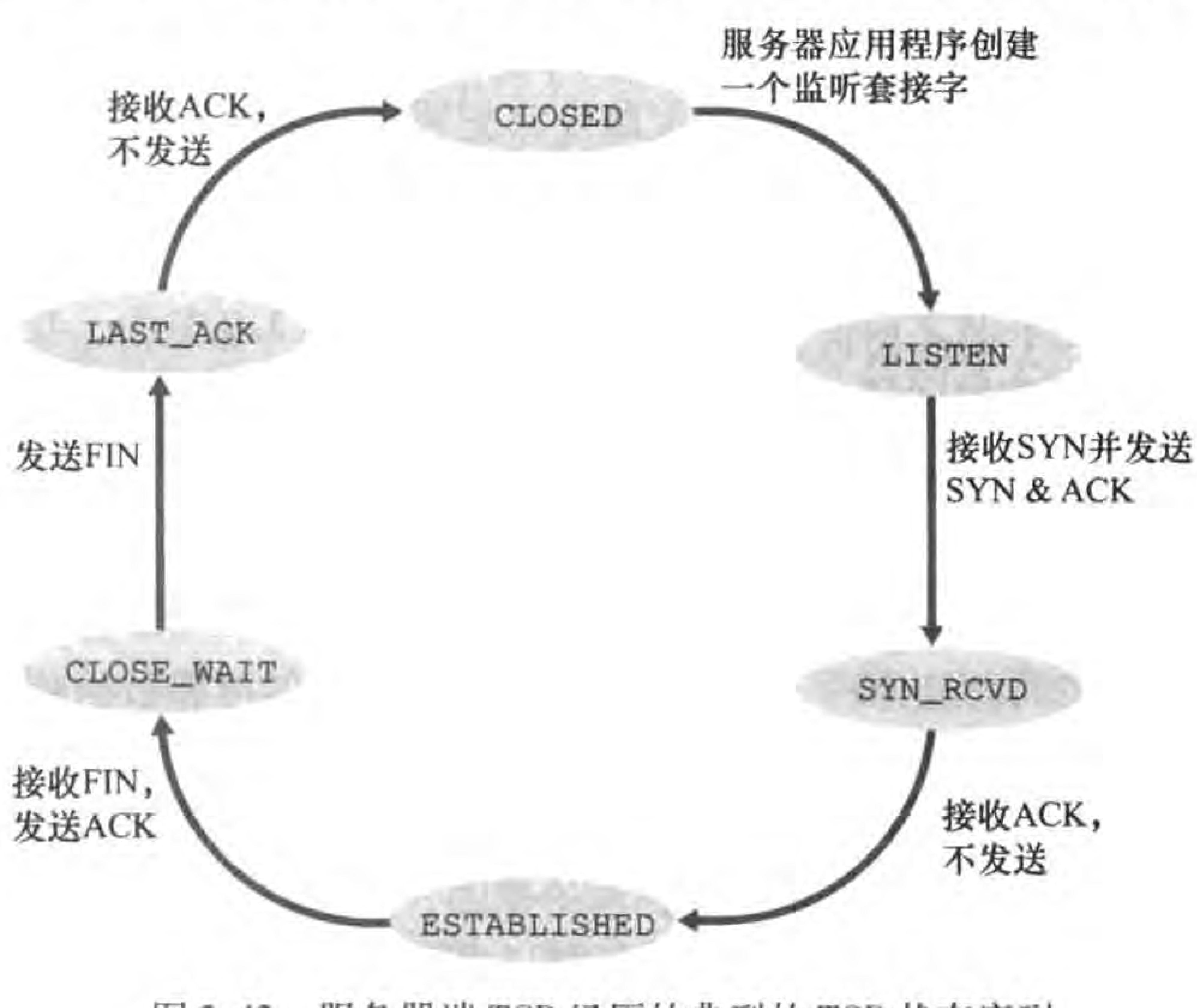


图 3-42 服务器端 TCP 经历的典型的 TCP 状态序列

除。这些状态变迁是自解释的。在这两个状态变迁图中，我们只给出了TCP连接是如何正常地被建立和拆除的。我们没有描述在某些不正常的情况下（例如当连接的双方同时都要发起或终止一条连接时）发生的事情。如果你对此问题及其他与TCP有关的高级问题感兴趣，推荐阅读Stevens的内容更全面的书籍［Stevens 1994］。

我们上面的讨论假定了客户和服务器都准备通信，即服务器正在监听客户发送其SYN报文段的端口。我们来考虑当一台主机接收到一个TCP报文段，其端口号或源IP地址与该主机上进行中的套接字都不匹配的情况。例如，假如一台主机接收了具有目的端口80的一个TCP SYN分组，但该主机在端口80不接受连接（即它不在端口80上运行Web服务器）。则该主机将向源发送一个特殊重置报文段。该TCP报文段将RST标志位（参见3.5.2节）置为1。因此，当主机发送一个重置报文段时，它告诉该源“我没有那个报文段的套接字。请不要再发送该报文段了”。当一台主机接收一个UDP分组，它的目的端口与进行中的UDP套接字不匹配，该主机发送一个特殊的ICMP数据报，这将在第4章中讨论。

既然我们已经对TCP连接管理有了深入的了解，我们再次回顾nmap端口扫描工具，并更为详细地研究它的工作原理。为了探索目标主机上的一个特定的TCP端口，如端口6789，nmap将对那台主机的目的端口6789发送一个特殊的TCP SYN报文段。有3种可能的输出：

- *源主机从目标主机接收到一个TCP SYNACK报文段。*因为这意味着在目标主机上一个应用程序使用TCP端口6789运行，nmap返回“打开”。
- *源主机从目标主机接收到一个TCP RST报文段。*这意味着该SYN报文段到达了目标主机，但目标主机没有运行一个使用TCP端口6789的应用程序。但攻击者至少知道发向该主机端口6789的报文段没有被源和目标主机之间的任何防火墙所阻挡。（将在第8章中讨论防火墙。）
- *源什么也没有收到。*这很可能表明该SYN报文段被中间的防火墙所阻挡，无法到达目标主机。

nmap是一个功能强大的工具，该工具不仅能“侦察”打开的TCP端口，也能“侦察”打开的UDP端口，还能“侦察”防火墙及其配置，甚至能“侦察”应用程序的版本和操作系统。其中的大多数都能通过操作TCP连接管理报文段完成［Skoudis 2006］。读者能够从www.nmap.org下载nmap。

到此，我们介绍完了TCP中的差错控制和流量控制。在3.7节中，我们将回到TCP并更深入地研究TCP拥塞控制问题。然而，在此之前，我们先后退一步，在更广泛环境中讨论拥塞控制问题。

3.6 拥塞控制原理

在前面几节中，我们已经分析了面临分组丢失时用于提供可靠数据传输服务的基本原理及特定的TCP机制。我们以前讲过，在实践中，这种丢包一般是当网络变得拥塞时由于路由器缓存溢出引起的。分组重传因此作为网络拥塞的征兆（某个特定的运输层报文段的丢失）来对待，但是却无法处理导致网络拥塞的原因，因为有太多的源想以过高的速率发送数据。为了处理网络拥塞原因，需要一些机制以在面临网络拥塞时遏制发送方。

在本节中，我们考虑一般情况下的拥塞控制问题，试图理解为什么网络拥塞是一件坏

事情，网络拥塞是如何在上层应用得到的服务性能中明确地显露出来的？如何可用各种方法来避免网络拥塞或对它做出反应？这种对拥塞控制的更一般研究是恰当的，因为就像可靠数据传输一样，它在组网技术中的前 10 个基础性重要问题清单中位居前列。下面一节详细研究 TCP 的拥塞控制算法。

3.6.1 拥塞原因与代价

我们通过分析 3 个复杂性越来越高的发生拥塞的情况，开始对拥塞控制的一般性研究。在每种情况下，我们首先将看看出现拥塞的原因以及拥塞的代价（根据资源未被充分利用以及端系统得到的低劣服务性能来评价）。我们暂不关注如何对拥塞做出反应或避免拥塞，而是重点理解一个较为简单的问题，即随着主机增加其发送速率并使网络变得拥塞，这时会发生的情况。

1. 情况 1：两个发送方和一台具有无穷大缓存的路由器

我们先考虑也许是最简单的拥塞情况：两台主机（A 和 B）都有一条连接，且这两条连接共享源与目的地之间的单跳路由，如图 3-43 所示。

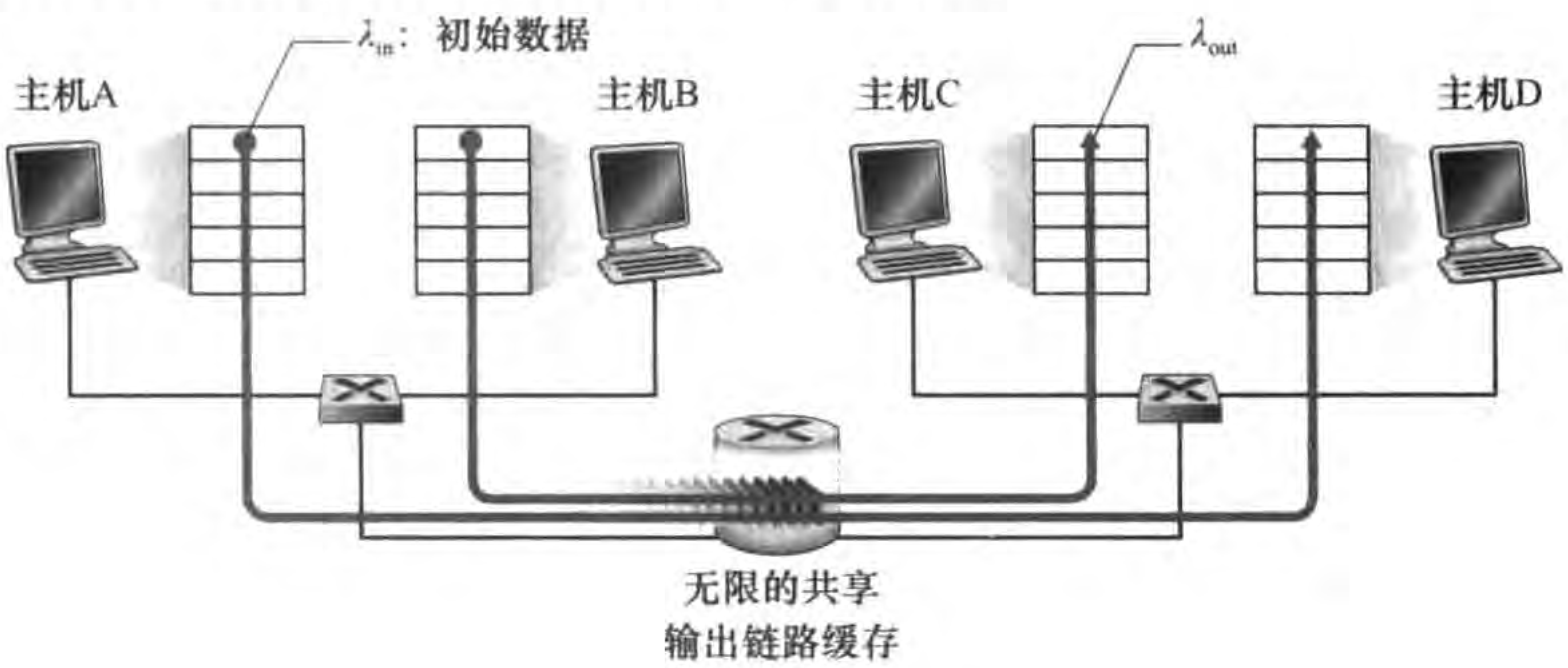


图 3-43 拥塞情况 1：两条连接共享具有无限大缓存的单跳路由

我们假设主机 A 中的应用程序以 λ_{in} 字节/秒的平均速率将数据发送到连接中（例如，通过一个套接字将数据传递给运输层协议）。这些数据是初始数据，这意味着每个数据单元仅向套接字中发送一次。下面的运输层协议是一个简单的协议。数据被封装并发送；不执行差错恢复（如重传）、流量控制或拥塞控制。忽略由于添加运输层和较低层首部信息产生的额外开销，在第一种情况下，主机 A 向路由器提供流量的速率是 λ_{in} 字节/秒。主机 B 也以同样的方式运行，为了简化问题，我们假设它也是以速率 λ_{in} 字节/秒发送数据。来自主机 A 和主机 B 的分组通过一台路由器，在一段容量为 R 的共享式输出链路上传输。该路由器带有缓存，可用于当分组到达速率超过该输出链路的容量时存储“入分组”。在此第一种情况下，我们将假设路由器有无限大的缓存空间。

图 3-44 描绘出了第一种情况下主机 A 的连接性能。左边的图形描绘了**每连接的吞吐量**（per-connection throughput）（接收方每秒接收的字节数）与该连接发送速率之间的函数关系。当发送速率在 0 ~ $R/2$ 之间时，接收方的吞吐量等于发送方的发送速率，即发送方发送的所有数据经有限时延后到达接收方。然而当发送速率超过 $R/2$ 时，它的吞吐量只能达 $R/2$。这个吞吐量上限是由两条连接之间共享链路容量造成的。链路完全不能以超过 $R/2$ 的稳定状态速率向接收方交付分组。无论主机 A 和主机 B 将其发送速率设置为多高，它们都不会看到超过 $R/2$ 的吞吐量。

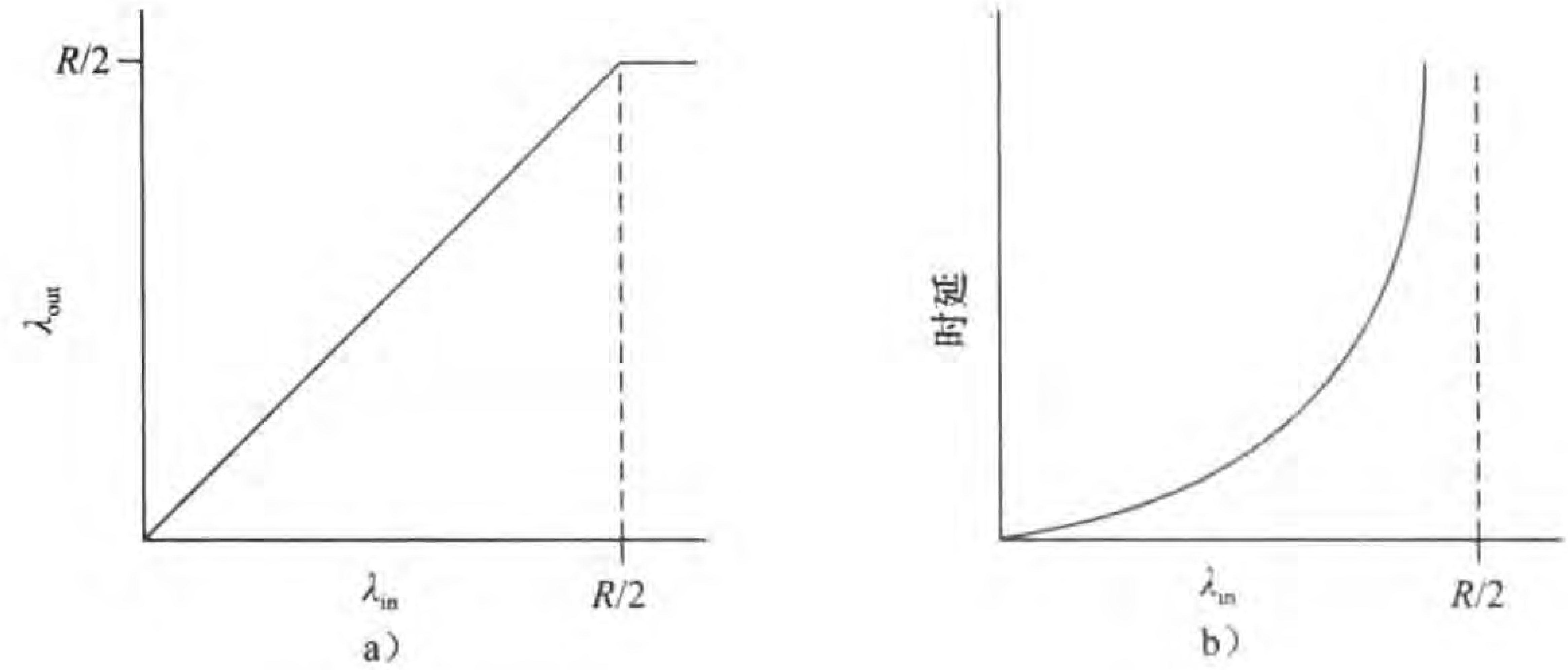


图 3-44 拥塞情况 1：吞吐量、时延与主机发送速率的函数关系

取得每连接 $R/2$ 的吞吐量实际上看起来可能是件好事，因为在将分组交付到目的地的过程中链路被充分利用了。但是，图 3-44b 的图形却显示了以接近链路容量的速率运行时产生的后果。当发送速率接近 $R/2$ 时（从左至右），平均时延就会越来越大。当发送速率超过 $R/2$ 时，路由器中的平均排队分组数就会无限增长，源与目的地之间的平均时延也会变成无穷大（假设这些连接以此发送速率运行无限长时间并且有无限量的缓存可用）。因此，虽然从吞吐量角度看，运行在总吞吐量接近 R 的状态也许是一个理想状态，但从时延角度看，却远不是一个理想状态。甚至在这种（极端）理想化的情况中，我们已经发现了拥塞网络的一种代价，即当分组的到达速率接近链路容量时，分组经历巨大的排队时延。

2. 情况2：两个发送方和一台具有有限缓存的路由器

现在我们从下列两个方面对情况 1 稍微做一些修改（参见图 3-45）。首先，假定路由器缓存的容量是有限的。这种现实世界的假设的结果是，当分组到达一个已满的缓存时会被丢弃。其次，我们假定每条连接都是可靠的。如果一个包含有运输层报文段的分组在路由器中被丢弃，那么它终将被发送方重传。由于分组可以被重传，所以我们现在必须更小心地使用发送速率这个术语。特别是我们再次以 λ_{in} 字节/秒表示应用程序将初始数据发送到套接字中的速率。运输层向网络中发送报文段（含有初始数据或重传数据）的速率用 λ'_{in} 字节/秒表示。λ'_{in} 有时被称为网络的**供给载荷**（offered load）。

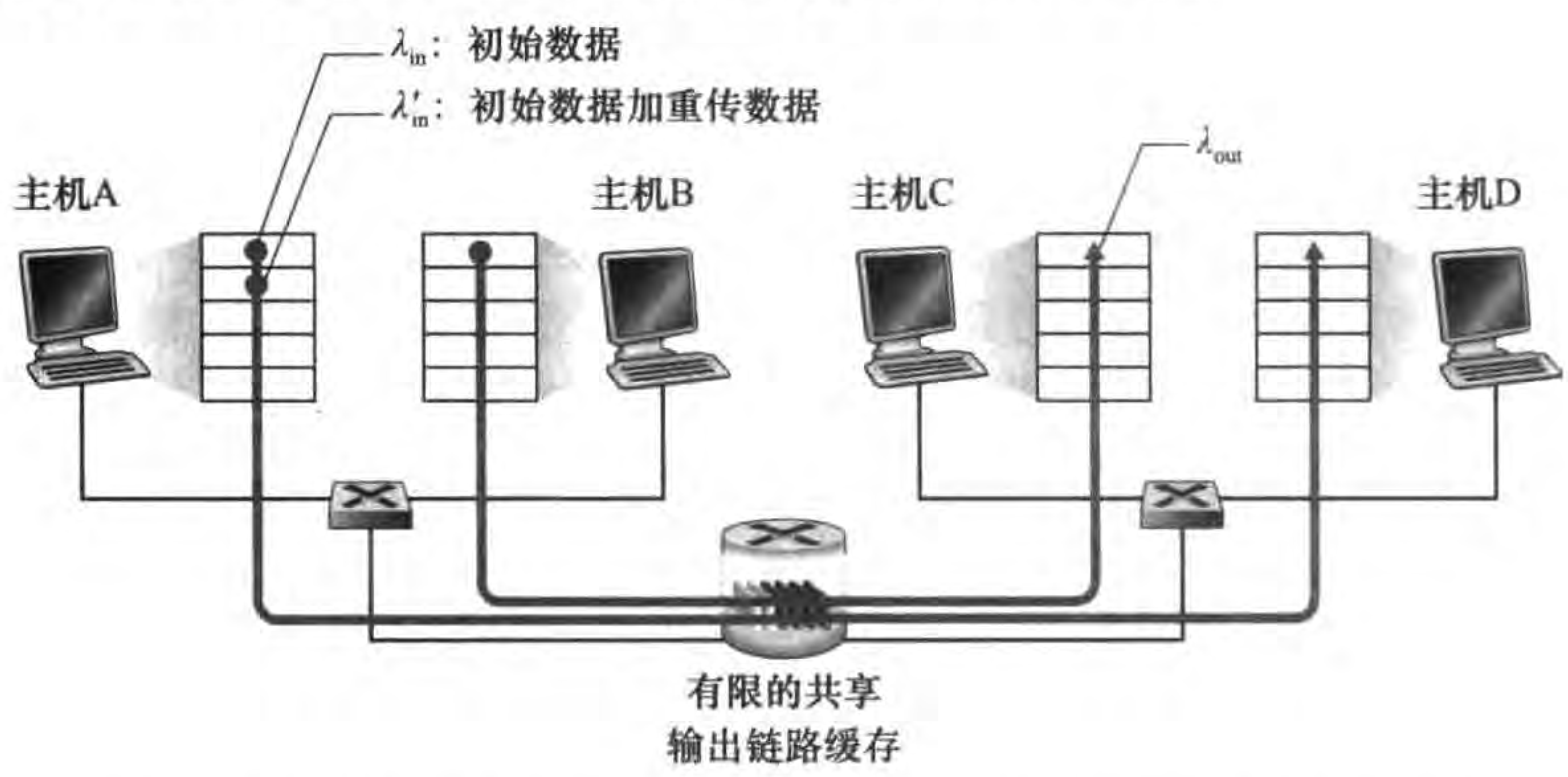


图 3-45 情况 2：（有重传的）两台主机与一台拥有有限缓存的路由器

在情况 2 下实现的性能强烈地依赖于重传的方式。首先，考虑一种不真实的情况，即

主机 A 能够以某种方式（不可思议地！）确定路由器中的缓存是否空闲，因而仅当缓存空闲时才发送一个分组。在这种情况下，将不会产生丢包，λ_{in}与λ'_{in}相等，并且连接的吞吐量就等于λ_{in}。图 3-46a 中描述了这种情况。从吞吐量的角度看，性能是理想的，即发送的每个分组都被接收到。注意到在这种情况下，平均主机发送速率不能超过 $R/2$，因为假定不会发生分组丢失。

接下来考虑一种更为真实的情况，发送方仅当在确定了一个分组已经丢失时才重传。（同样，所做的假设有一些弹性。然而，发送主机有可能将超时时间设置得足够长，以无形中使其确信一个还没有被确认的分组已经丢失。）在这种情况下，性能就可能与图 3-46b 所示的情况相似。为了理解这时发生的情况，考虑一下供给载荷λ'_{in}（初始数据传输加上重传的总速率）等于 $R/2$ 的情况。根据图 3-46b，在这一供给载荷值时，数据被交付给接收方应用程序的速率是 $R/3$。因此，在所发送的 0.5R 单位数据当中，从平均的角度说，0.333R 字节/秒是初始数据，而 0.166R 字节/秒是重传数据。*我们在此看到了另一种网络拥塞的代价，即发送方必须执行重传以补偿因为缓存溢出而丢弃（丢失）的分组。*

最后，我们考虑下面一种情况：发送方也许会提前发生超时并重传在队列中已被推迟但还未丢失的分组。在这种情况下，初始数据分组和重传分组都可能到达接收方。当然，接收方只需要一份这样的分组副本就行了，重传分组将被丢弃。在这种情况下，路由器转发重传的初始分组副本是在做无用功，因为接收方已收到了该分组的初始版本。而路由器本可以利用链路的传输能力去发送另一个分组。*这里，我们又看到了网络拥塞的另一种代价，即发送方在遇到大时延时所进行的不必要重传会引起路由器利用其链路带宽来转发不必要的分组副本。*图 3-46c 显示了当假定每个分组被路由器转发（平均）两次时，吞吐量与供给载荷的对比情况。由于每个分组被转发两次，当其供给载荷接近 $R/2$ 时，其吞吐量将渐近 $R/4$。

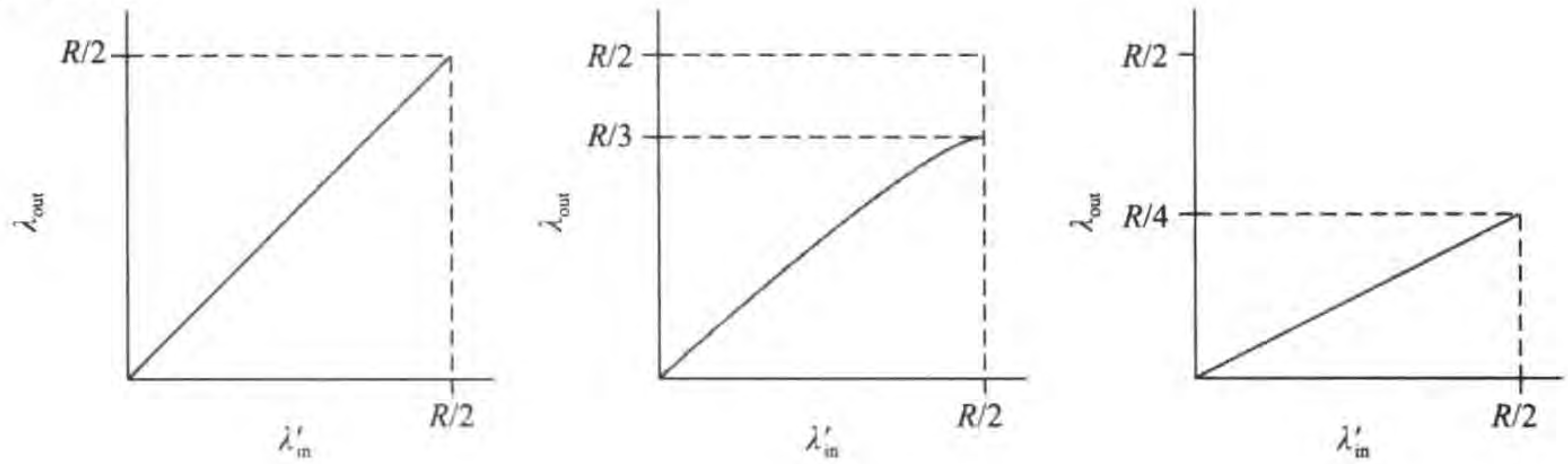


图 3-46 具有有限缓存时情况 2 的性能

3. 情况 3：4 个发送方和具有有限缓存的多台路由器及多跳路径

在最后一种拥塞情况中，有 4 台主机发送分组，每台都通过交叠的两跳路径传输，如图 3-47 所示。我们再次假设每台主机都采用超时/重传机制来实现可靠数据传输服务，所有的主机都有相同的λ_{in}值，所有路由器的链路容量都是 R 字节/秒。

我们考虑从主机 A 到主机 C 的连接，该连接经过路由器 R1 和 R2。A－C 连接与 D－B 连接共享路由器 R1，并与 B－D 连接共享路由器 R2。对极小的λ_{in}值，路由器缓存的溢出是很少见的（与拥塞情况 1、拥塞情况 2 中的一样），吞吐量大致接近供给载荷。对稍大的λ_{in}值，对应的吞吐量也更大，因为有更多的初始数据被发送到网络中并交付到目的地，溢出仍然很少。因此，对于较小的λ_{in}，λ_{in}的增大会导致λ_{out}的增大。

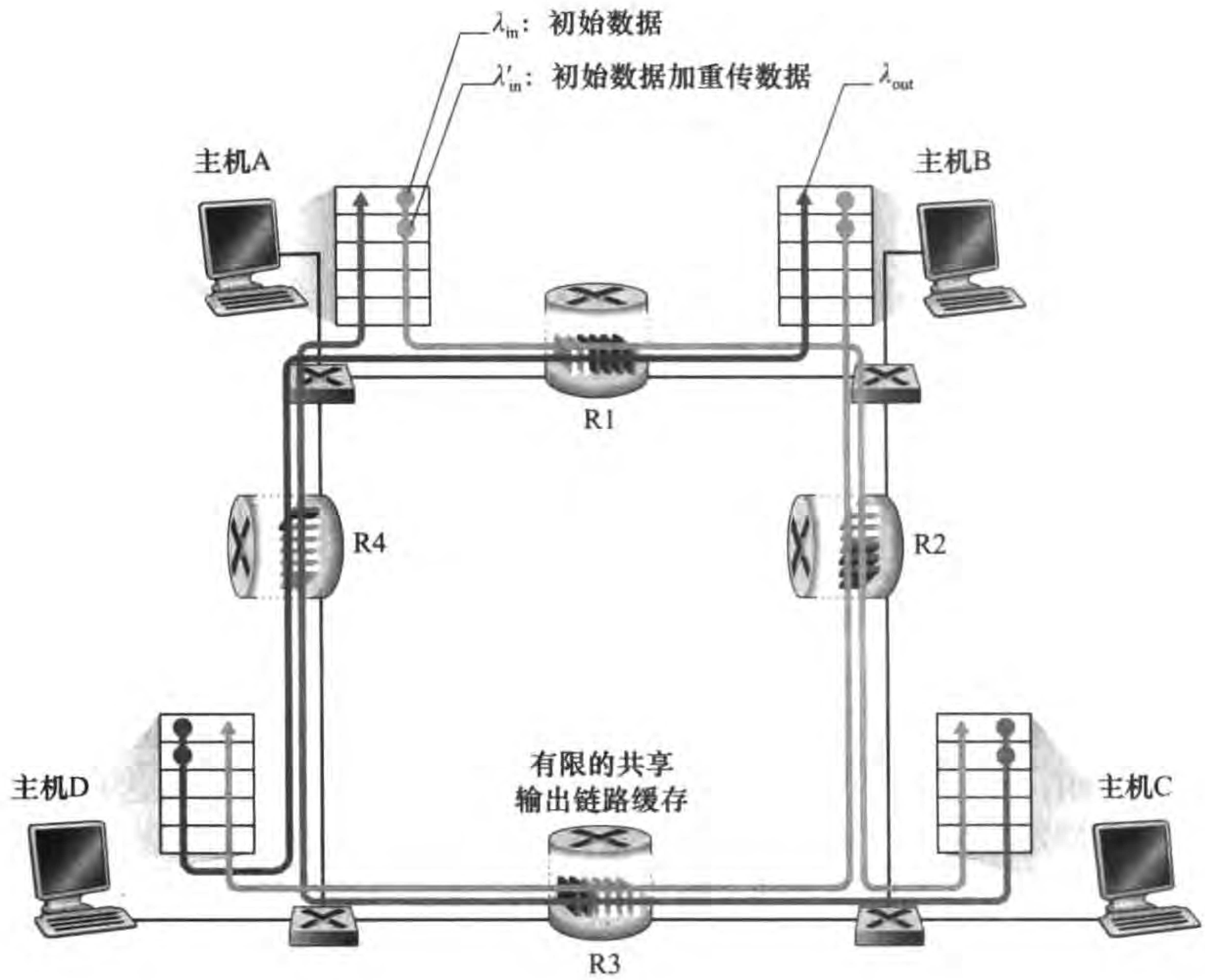


图 3-47　四个发送方和具有有限缓存的多台路由器及多跳路径

在考虑了流量很小的情况后，下面分析当 λ_{in}（因此 λ'_{in}）很大时的情况。考虑路由器 R2。不管 λ_{in}的值是多大，到达路由器 R2 的 A－C 流量（在经过路由器 R1 转发后到达路由器 R2）的到达速率至多是 R，也就是从 R1 到 R2 的链路容量。如果 λ'_{in}对于所有连接（包括 B－D 连接）来说是极大的值，那么在 R2 上，B－D 流量的到达速率可能会比 A－C 流量的到达速率大得多。因为 A－C 流量与 B－D 流量在路由器 R2 上必须为有限缓存空间而竞争，所以当来自 B－D 连接的供给载荷越来越大时，A－C 连接上成功通过 R2（即由于缓存溢出而未被丢失）的流量会越来越小。在极限情况下，当供给载荷趋近于无穷大时，R2 的空闲缓存会立即被 B－D 连接的分组占满，因而A－C 连接在 R2 上的吞吐量趋近于 0。这又一次说明在重载的极限情况下，A－C 端到端吞吐量将趋近于 0。这些考虑引发了供给载荷与吞吐量之间的权衡，如图 3-48 所示。

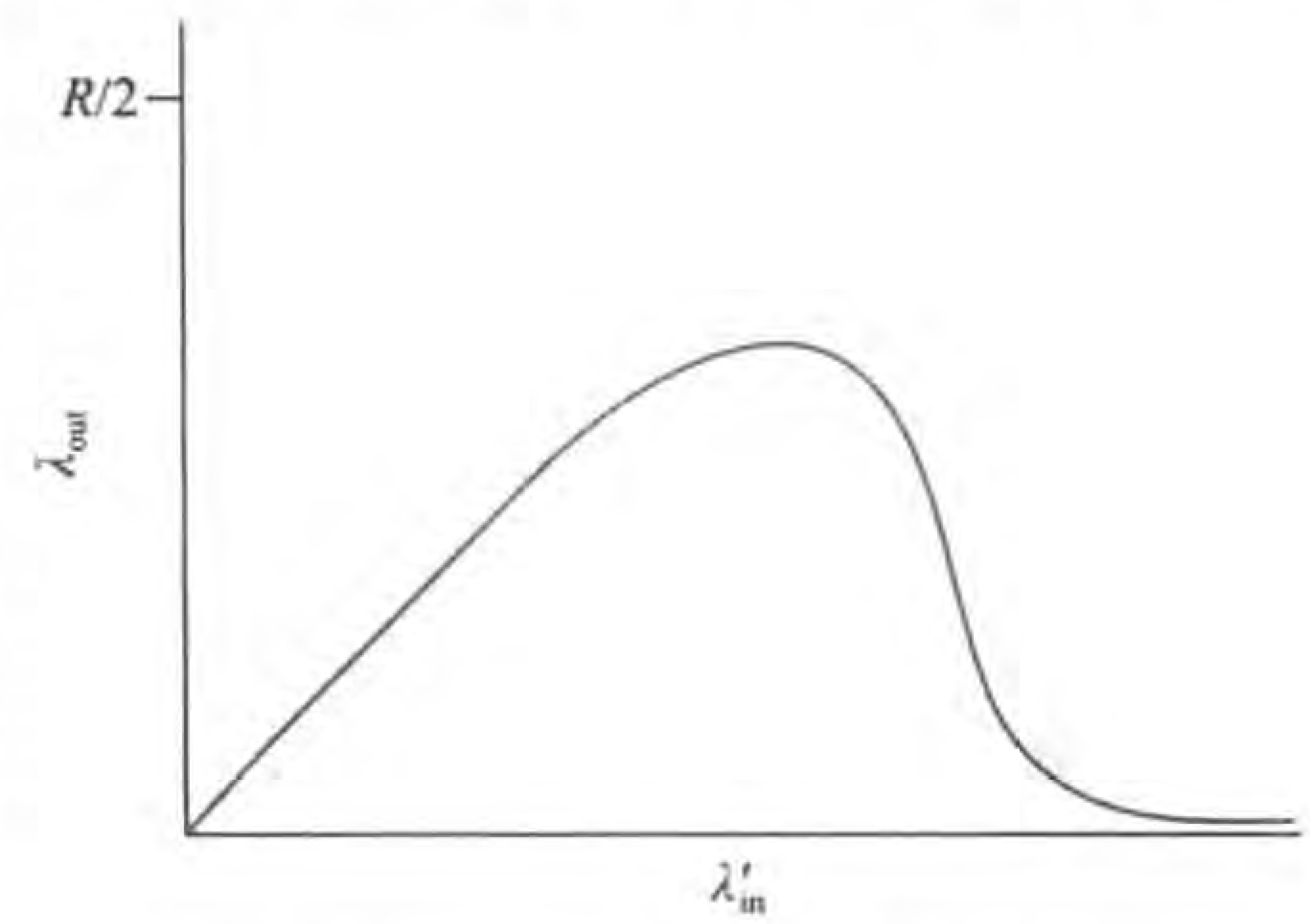


图 3-48　具有有限缓存和多跳路径时的情况 3 性能

当考虑由网络所做的浪费掉的工作量时，随着供给载荷的增加而使吞吐量最终减少的原因是明显的。在上面提到的大流量的情况中，每当有一个分组在第二跳路由器上被丢弃时，第一跳路由器所做的将分组转发到第二跳路由器的工作就是“劳而无功”的。如果第一跳路由器只是丢弃该分组并保持空闲，则网络中的情况是幸运的（更准确地说是糟糕的）。需要指出的是，第一跳路由器所使用的将分组转发到第二跳路由器的传输容量用来传送不同的分组可能更有效

益。（例如，当选择一个分组发送时，路由器最好优先考虑那些已经历过一定数量的上游路由器的分组。）所以，我们在此又看到了由于拥塞而丢弃分组的另一种代价，即当一个分组沿一条路径被丢弃时，每个上游路由器用于转发该分组到丢弃该分组而使用的传输容量最终被浪费掉了。

3.6.2 拥塞控制方法

在3.7节中，我们将详细研究TCP用于拥塞控制的特定方法。这里，我们指出在实践中所采用的两种主要拥塞控制方法，讨论特定的网络体系结构和具体使用这些方法的拥塞控制协议。

在最为宽泛的级别上，我们可根据网络层是否为运输层拥塞控制提供了显式帮助，来区分拥塞控制方法。

- *端到端拥塞控制*。在端到端拥塞控制方法中，网络层没有为运输层拥塞控制提供显式支持。即使网络中存在拥塞，端系统也必须通过对网络行为的观察（如分组丢失与时延）来推断之。我们将在3.7.1节中将看到，TCP采用端到端的方法解决拥塞控制，因为IP层不会向端系统提供有关网络拥塞的反馈信息。TCP报文段的丢失（通过超时或3次冗余确认而得知）被认为是网络拥塞的一个迹象，TCP会相应地减小其窗口长度。我们还将看到关于TCP拥塞控制的一些最新建议，即使用增加的往返时延值作为网络拥塞程度增加的指示。
- *网络辅助的拥塞控制*。在网络辅助的拥塞控制中，路由器向发送方提供关于网络中拥塞状态的显式反馈信息。这种反馈可以简单地用一个比特来指示链路中的拥塞情况。该方法在早期的IBM SNA [Schwartz 1982]、DEC DECnet [Jain 1989; Ramakrishnan 1990] 和ATM [Black 1995] 等体系结构中被采用。更复杂的网络反馈也是可能的。例如，在ATM **可用比特率**（Available Bite Rate，ABR）拥塞控制中，路由器显式地通知发送方它（路由器）能在输出链路上支持的最大主机发送速率。如上面所提到的，默认因特网版本的IP和TCP采用端到端拥塞控制方法。然而，我们在3.7.2节中将看到，最近IP和TCP也能够选择性地实现网络辅助拥塞控制。

对于网络辅助的拥塞控制，拥塞信息从网络反馈到发送方通常有两种方式，如图3-49

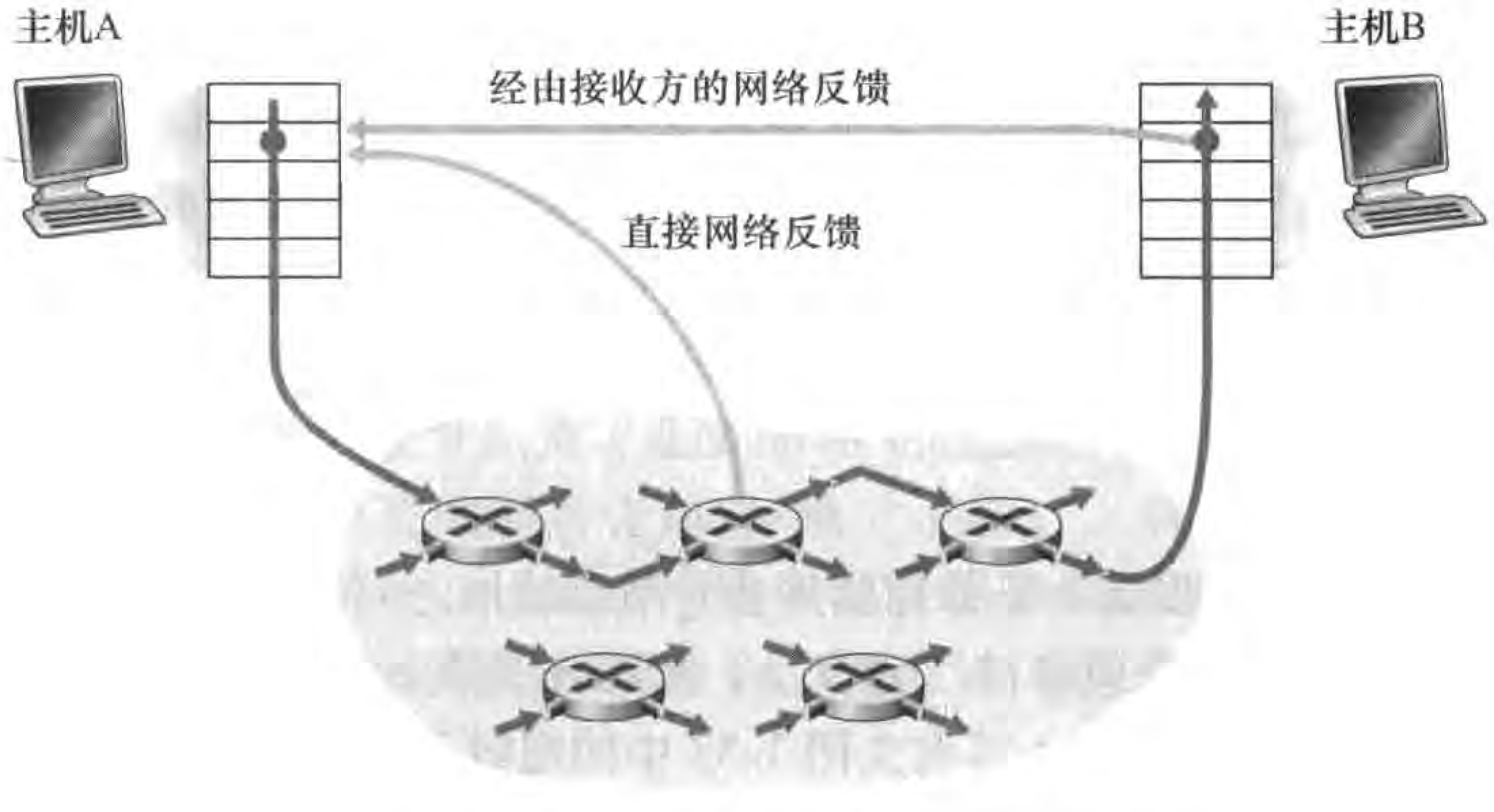


图3-49　网络指示拥塞信息的两种反馈路径

所示。直接反馈信息可以由网络路由器发给发送方。这种方式的通知通常采用了一种**阻塞分组**（choke packet）的形式（主要是说："我拥塞了！"）。更为通用的第二种形式的通知是，路由器标记或更新从发送方流向接收方的分组中的某个字段来指示拥塞的产生。一旦收到一个标记的分组后，接收方就会向发送方通知该网络拥塞指示。注意到后一种形式的通知至少要经过一个完整的往返时间。

3.7 TCP 拥塞控制

在本节中，我们再次来学习 TCP。如我们在 3.5 节所见，TCP 为运行在不同主机上的两个进程之间提供了可靠传输服务。TCP 的另一个关键部分就是其拥塞控制机制。如在前一节所指出，TCP 必须使用端到端拥塞控制而不是使网络辅助的拥塞控制，因为 IP 层不向端系统提供显式的网络拥塞反馈。

TCP 所采用的方法是让每一个发送方根据所感知到的网络拥塞程度来限制其能向连接发送流量的速率。如果一个 TCP 发送方感知从它到目的地之间的路径上没什么拥塞，则 TCP 发送方增加其发送速率；如果发送方感知沿着该路径有拥塞，则发送方就会降低其发送速率。但是这种方法提出了三个问题。第一，一个 TCP 发送方如何限制它向其连接发送流量的速率呢？第二，一个 TCP 发送方如何感知从它到目的地之间的路径上存在拥塞呢？第三，当发送方感知到端到端的拥塞时，采用何种算法来改变其发送速率呢？

我们首先分析一下 TCP 发送方是如何限制向其连接发送流量的。在 3.5 节中我们看到，TCP 连接的每一端都是由一个接收缓存、一个发送缓存和几个变量（LastByteRead、rwnd 等）组成。运行在发送方的 TCP 拥塞控制机制跟踪一个额外的变量，即**拥塞窗口**（congestion window）。拥塞窗口表示为 cwnd，它对一个 TCP 发送方能向网络中发送流量的速率进行了限制。特别是，在一个发送方中未被确认的数据量不会超过 cwnd 与 rwnd 中的最小值，即

$$\text{LastByteSent} - \text{LastByteAcked} \leq \min\{\text{cwnd}, \text{rwnd}\}$$

为了关注拥塞控制（与流量控制形成对比），我们后面假设 TCP 接收缓存足够大，以至可以忽略接收窗口的限制；因此在发送方中未被确认的数据量仅受限于 cwnd。我们还假设发送方总是有数据要发送，即在拥塞窗口中的所有报文段要被发送。

上面的约束限制了发送方中未被确认的数据量，因此间接地限制了发送方的发送速率。为了理解这一点，我们来考虑一个丢包和发送时延均可以忽略不计的连接。因此粗略地讲，在每个往返时间（RTT）的起始点，上面的限制条件允许发送方向该连接发送 cwnd 个字节的数据，在该 RTT 结束时发送方接收对数据的确认报文。因此，该发送方的发送速率大概是 cwnd/RTT 字节/秒。通过调节 cwnd 的值，发送方因此能调整它向连接发送数据的速率。

我们接下来考虑 TCP 发送方是如何感知在它与目的地之间的路径上出现了拥塞的。我们将一个 TCP 发送方的"丢包事件"定义为：要么出现超时，要么收到来自接收方的 3 个冗余 ACK。（回想我们在 3.5.4 节有关图 3-33 中的超时事件的讨论和收到 3 个冗余 ACK 后包括快速重传的后继修改。）当出现过度的拥塞时，在沿着这条路径上的一台（或多台）路由器的缓存会溢出，引起一个数据报（包含一个 TCP 报文段）被丢弃。丢弃的数

据报接着会引起发送方的丢包事件（要么超时或收到3个冗余ACK），发送方就认为在发送方到接收方的路径上出现了拥塞的指示。

考虑了拥塞检测问题后，我们接下来考虑网络没有拥塞这种更为乐观的情况，即没有出现丢包事件的情况。在此情况下，在TCP的发送方将收到对于以前未确认报文段的确认。如我们将看到的那样，TCP将这些确认的到达作为一切正常的指示，即在网络上传输的报文段正被成功地交付给目的地，并使用确认来增加窗口的长度（及其传输速率）。注意到如果确认以相当慢的速率到达（例如，如果该端到端路径具有高时延或包含一段低带宽链路），则该拥塞窗口将以相当慢的速率增加。在另一方面，如果确认以高速率到达，则该拥塞窗口将会更为迅速地增大。因为TCP使用确认来触发（或计时）增大它的拥塞窗口长度，TCP被说成是**自计时**（self-clocking）的。

给定调节cwnd值以控制发送速率的机制，关键的问题依然存在：TCP发送方怎样确定它应当发送的速率呢？如果众多TCP发送方总体上发送太快，它们能够拥塞网络，导致我们在图3-48中看到的拥塞崩溃。事实上，为了应对在较早TCP版本下观察到的因特网拥塞崩溃［Jacobson 1988］，研发了该版本的TCP（我们马上将学习它）。然而，如果TCP发送方过于谨慎，发送太慢，它们不能充分利用网络的带宽；这就是说，TCP发送方能够以更高的速率发送而不会使网络拥塞。那么TCP发送方如何确定它们的发送速率，既使得网络不会拥塞，与此同时又能充分利用所有可用的带宽？TCP发送方是显式地协作，或存在一种分布式方法使TCP发送方能够仅基于本地信息设置它们的发送速率？TCP使用下列指导性原则回答这些问题：

- *一个丢失的报文段表意味着拥塞，因此当丢失报文段时应当降低TCP发送方的速率。*回想在3.5.4节中的讨论，对于给定报文段，一个超时事件或四个确认（一个初始ACK和其后的三个冗余ACK）被解释为跟随该四个ACK的报文段的“丢包事件”的一种隐含的指示。从拥塞控制的观点看，该问题是TCP发送方应当如何减小它的拥塞窗口长度，即减小其发送速率，以应对这种推测的丢包事件。
- *一个确认报文段指示该网络正在向接收方交付发送方的报文段，因此，当对先前未确认报文段的确认到达时，能够增加发送方的速率。*确认的到达被认为是一切顺利的隐含指示，即报文段正从发送方成功地交付给接收方，因此该网络不拥塞。拥塞窗口长度因此能够增加。
- *带宽探测。*给定ACK指示源到目的地路径无拥塞，而丢包事件指示路径拥塞，TCP调节其传输速率的策略是增加其速率以响应到达的ACK，除非出现丢包事件，此时才减小传输速率。因此，为探测拥塞开始出现的速率，TCP发送方增加它的传输速率，从该速率后退，进而再次开始探测，看看拥塞开始速率是否发生了变化。TCP发送方的行为也许类似于要求（并得到）越来越多糖果的孩子，直到最后告知他/她“不行!”，孩子后退一点，然后过一会儿再次开始提出请求。注意到网络中没有明确的拥塞状态信令，即ACK和丢包事件充当了隐式信号，并且每个TCP发送方根据异步于其他TCP发送方的本地信息而行动。

概述了TCP拥塞控制后，现在是我们考虑广受赞誉的**TCP拥塞控制算法**（TCP congestion control algorithm）细节的时候了，该算法首先在［Jacobson 1988］中描述并

且在［RFC 5681］中标准化。该算法包括3个主要部分：①慢启动；②拥塞避免；③快速恢复。慢启动和拥塞避免是TCP的强制部分，两者的差异在于对收到的ACK做出反应时增加cwnd长度的方式。我们很快将会看到慢启动比拥塞避免能更快地增加cwnd的长度（不要被名称所迷惑！）。快速恢复是推荐部分，对TCP发送方并非是必需的。

1. 慢启动

当一条TCP连接开始时，cwnd的值通常初始置为一个MSS的较小值［RFC 3390］，这就使得初始发送速率大约为MSS/RTT。例如，如果MSS = 500字节且RTT = 200ms，则得到的初始发送速率大约只有20kbps。由于对TCP发送方而言，可用带宽可能比MSS/RTT大得多，TCP发送方希望迅速找到可用带宽的数量。因此，在**慢启动**（slow-start）状态，cwnd的值以1个MSS开始并且每当传输的报文段首次被确认就增加1个MSS。在图3-50所示的例子中，TCP向网络发送第一个报文段并等待一个确认。当该确认到达时，TCP发送方将拥塞窗口增加一个MSS，并发送出两个最大长度的报文段。这两个报文段被确认，则发送方对每个确认报文段将拥塞窗口增加一个MSS，使得拥塞窗口变为4个MSS，并这样下去。这一过程每过一个RTT，发送速率就翻番。因此，TCP发送速率起始慢，但在慢启动阶段以指数增长。

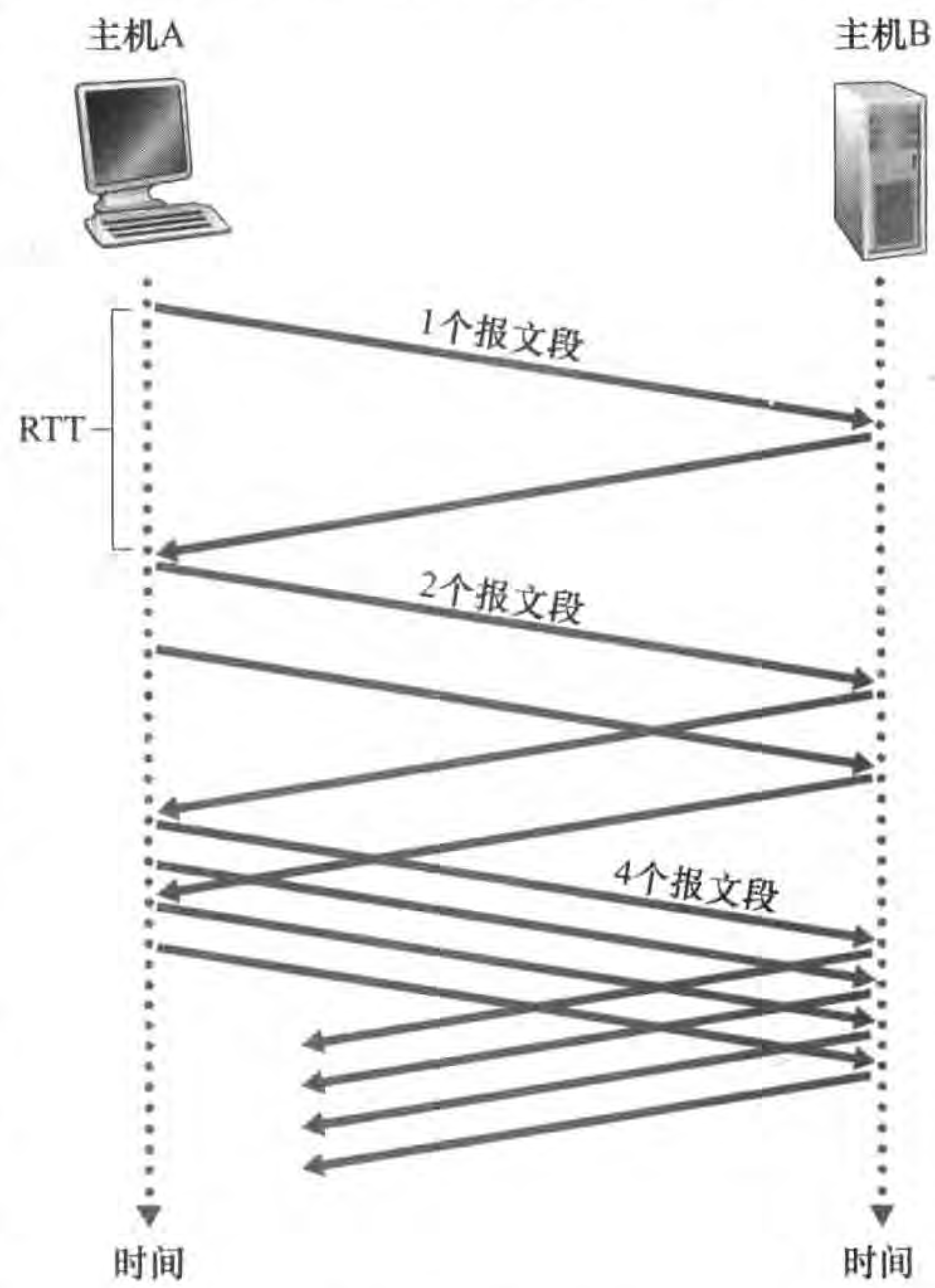


图3-50 TCP慢启动

但是，何时结束这种指数增长呢？慢启动对这个问题提供了几种答案。首先，如果存在一个由超时指示的丢包事件（即拥塞），TCP发送方将cwnd设置为1并重新开始慢启动过程。它还将第二个状态变量的值ssthresh（“慢启动阈值”的速记）设置为cwnd/2，即当检测到拥塞时将ssthresh置为拥塞窗口值的一半。慢启动结束的第二种方式是直接与ssthresh的值相关联。因为当检测到拥塞时ssthresh设为cwnd的值一半，当到达或超过ssthresh的值时，继续使cwnd翻番可能有些鲁莽。因此，当cwnd的值等于ssthresh时，结束慢启动并且TCP转移到拥塞避免模式。我们将会看到，当进入拥塞避免模式时，TCP更为谨慎地增加cwnd。最后一种结束慢启动的方式是，如果检测到3个冗余ACK，这时TCP执行一种快速重传（参见3.5.4节）并进入快速恢复状态，后面将讨论相关内容。慢启动中的TCP行为总结在图3-51中的TCP拥塞控制的FSM描述中。慢启动算法最早源于［Jacobson 1988］；在［Jain 1986］中独立地提出了一种类似于慢启动的方法。

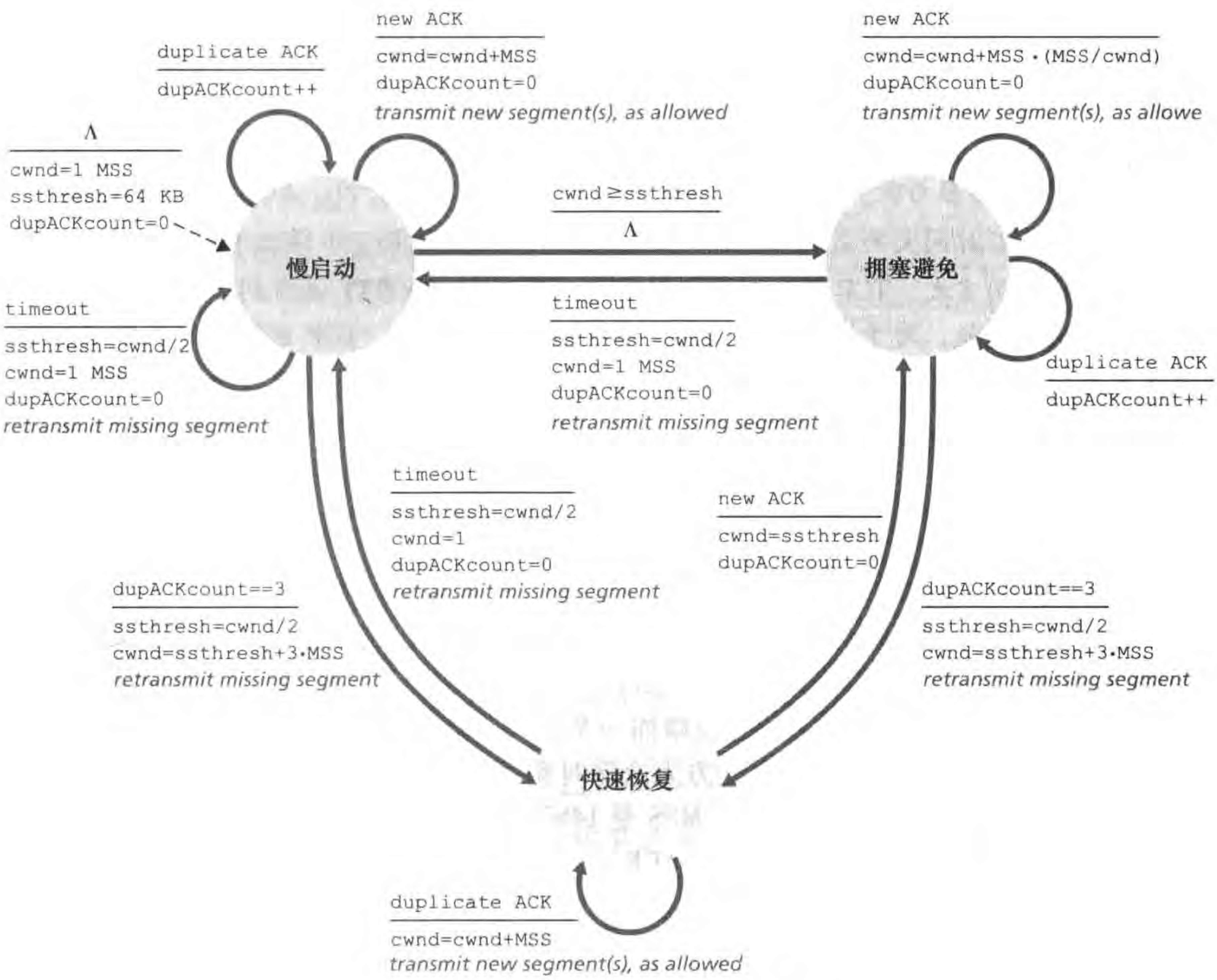


图 3-51 TCP 拥塞控制的 FSM 描述

实践原则

TCP 分岔：优化云服务的性能

对于诸如搜索、电子邮件和社交网络等云服务，非常希望提供高水平的响应性，给用户一种完美的印象，即这些服务正运行在它们自己的端系统（包括其智能手机）中。因为用户经常位于远离数据中心的地方，而这些数据中心负责为云服务关联的动态内容提供服务。实际上，如果端系统远离数据中心，则 RTT 将会很大，会由于 TCP 慢启动潜在地导致低劣的响应时间性能。

作为一个学习案例，考虑接收对某搜索问题的响应中的时延。通常，服务器在慢启动期间交付响应要求三个 TCP 窗口 [Pathak 2010]。所以从某端系统发起一条 TCP 连接到它收到该响应的最后一个分组的时间粗略是 4 * RTT（用于建立 TCP 连接的一个 RTT 加上用于 3 个数据窗口的 3 个 RTT），再加上在数据中心中处理的时间。对于一个相当小的查询来说，这些 RTT 时延导致其返回搜索结果中显而易见的时延。此外，在接入网中可能有较大的丢包，导致 TCP 重传甚至较大的时延。

缓解这个问题和改善用户感受到的性能的一个途径是：①部署邻近用户的前端服务器；②在该前端服务器利用 **TCP 分岔**（TCP splitting）来分裂 TCP 连接。借助于 TCP 分

岔，客户向邻近前端连接一条 TCP 连接，并且该前端以非常大的窗口向数据中心维护一条 TCP 连接［Tariq 2008，Pathak 2010，Chen 2011］。使用这种方法，响应时间大致变为 $4 * RTT_{FE} + RTT_{BE}$ + 处理时间，其中 RTT_{FE} 是客户与前端服务器之间的往返时间，RTT_{BE} 是前端服务器与数据中心（后端服务器）之间的往返时间。如果前端服务器邻近客户，则该响应时间大约变为 RTT_{BE} 加上处理时间，因为 RTT_{FE} 小得微不足道并且 RTT_{BE} 约为 RTT。总而言之，TCP 分岔大约能够将网络时延从 4 * RTT 减少到 RTT，极大地改善用户感受的性能，对于远离最近数据中心的用户更是如此。TCP 分岔也有助于减少因接入网丢包引起的 TCP 重传时延。今天，Google 和 Akamai 在接入网中广泛利用了它们的 CDN 服务器（回想 2.6 节中的讨论），为它们支持的云服务来执行 TCP 分岔［Chen 2011］。

2. 拥塞避免

一旦进入拥塞避免状态，cwnd 的值大约是上次遇到拥塞时的值的一半，即距离拥塞可能并不遥远！因此，TCP 无法每过一个 RTT 再将 cwnd 的值翻番，而是采用了一种较为保守的方法，每个 RTT 只将 cwnd 的值增加一个 MSS［RFC 5681］。这能够以几种方式完成。一种通用的方法是对于 TCP 发送方无论何时到达一个新的确认，就将 cwnd 增加一个 MSS（MSS/cwnd）字节。例如，如果 MSS 是 1460 字节并且 cwnd 是 14 600 字节，则在一个 RTT 内发送 10 个报文段。每个到达 ACK（假定每个报文段一个 ACK）增加 1/10MSS 的拥塞窗口长度，因此在收到对所有 10 个报文段的确认后，拥塞窗口的值将增加了一个 MSS。

但是何时应当结束拥塞避免的线性增长（每 RTT 1MSS）呢？当出现超时时，TCP 的拥塞避免算法行为相同。与慢启动的情况一样，cwnd 的值被设置为 1 个 MSS，当丢包事件出现时，ssthresh 的值被更新为 cwnd 值的一半。然而，前面讲过丢包事件也能由一个三个冗余 ACK 事件触发。在这种情况下，网络继续从发送方向接收方交付报文段（就像由收到冗余 ACK 所指示的那样）。因此 TCP 对这种丢包事件的行为，相比于超时指示的丢包，应当不那么剧烈：TCP 将 cwnd 的值减半（为使测量结果更好，计及已收到的 3 个冗余的 ACK 要加上 3 个 MSS），并且当收到 3 个冗余的 ACK，将 ssthresh 的值记录为 cwnd 的值的一半。接下来进入快速恢复状态。

3. 快速恢复

在快速恢复中，对于引起 TCP 进入快速恢复状态的缺失报文段，对收到的每个冗余的 ACK，cwnd 的值增加一个 MSS。最终，当对丢失报文段的一个 ACK 到达时，TCP 在降低 cwnd 后进入拥塞避免状态。如果出现超时事件，快速恢复在执行如同在慢启动和拥塞避免中相同的动作后，迁移到慢启动状态：当丢包事件出现时，cwnd 的值被设置为 1 个 MSS，并且 ssthresh 的值设置为 cwnd 值的一半。

快速恢复是 TCP 推荐的而非必需的构件［RFC 5681］。有趣的是，一种称为 TCP Tahoe 的 TCP 早期版本，不管是发生超时指示的丢包事件，还是发生 3 个冗余 ACK 指示的丢包事件，都无条件地将其拥塞窗口减至 1 个 MSS，并进入慢启动阶段。TCP 的较新版本 TCP Reno，则综合了快速恢复。

图 3-52 图示了 Reno 版 TCP 与 Tahoe 版 TCP 的拥塞控制窗口的演化情况。在该图中，阈值初始等于 8 个 MSS。在前 8 个传输回合，Tahoe 和 Reno 采取了相同的动作。拥塞窗口在慢启动阶段以指数速度快速爬升，在第 4 轮传输时到达了阈值。然后拥塞窗口以线性速度爬升，直到在第 8 轮传输后出现 3 个冗余 ACK。注意到当该丢包事件发生时，拥塞窗口值为 12 × MSS。于是 ssthresh 的值被设置为 0.5 × cwnd = 6 × MSS。在 TCP Reno 下，拥塞窗口被设置为 cwnd = 9MSS，然后线性地增长。在 TCP Tahoe 下，拥塞窗口被设置为 1 个 MSS，然后呈指数增长，直至到达 ssthresh 值为止，在这个点它开始线性增长。

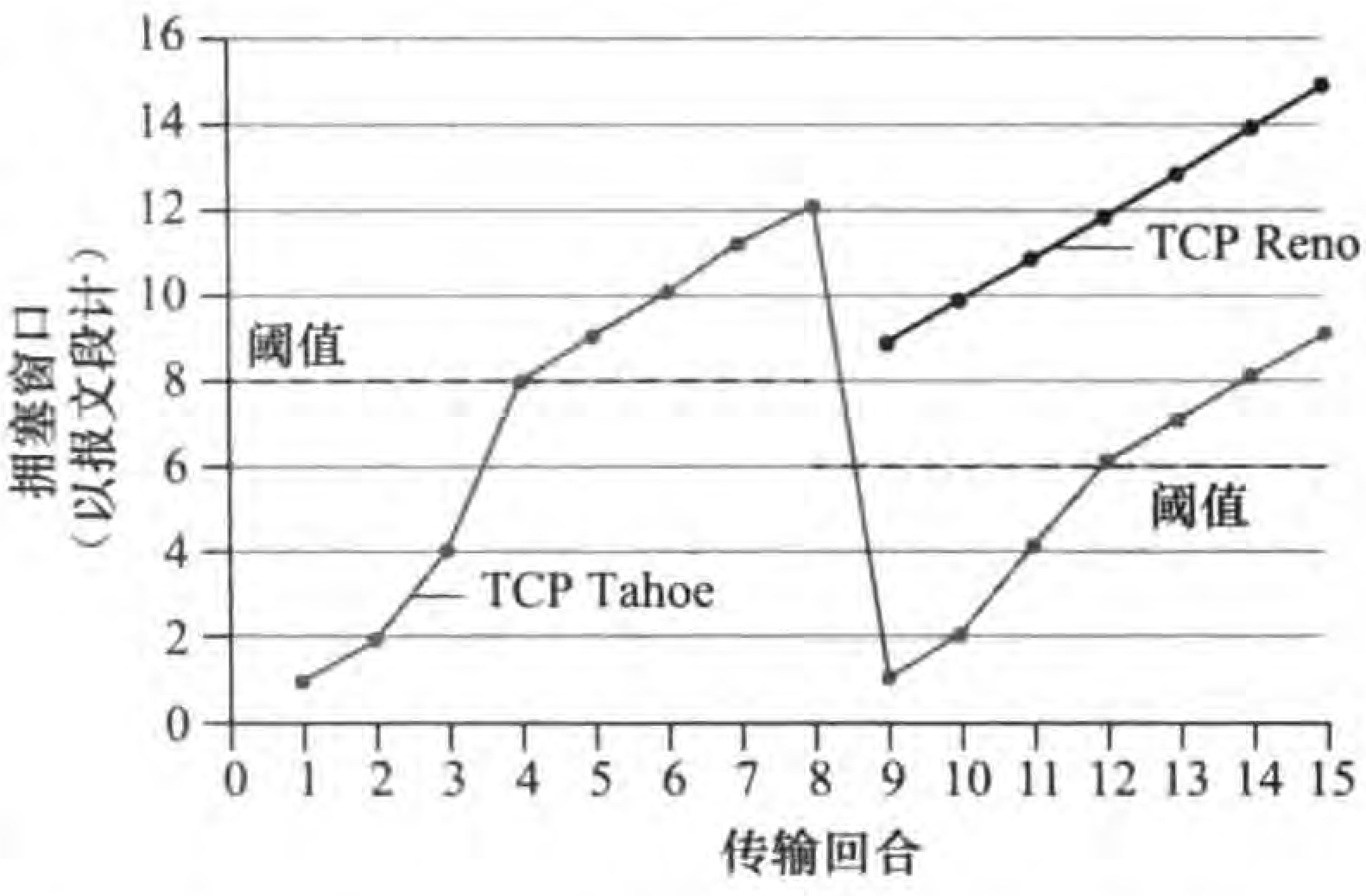


图 3-52 TCP 拥塞窗口的演化（Tahoe 和 Reno）

图 3-51 表示了 TCP 拥塞控制算法（即慢启动、拥塞避免和快速恢复）的完整 FSM 描述。该图也指示了新报文段的传输或重传的报文段可能出现的位置。尽管区分 TCP 差错控制/重传与 TCP 拥塞控制非常重要，但是注意到 TCP 这两个方面交织链接的方式也很重要。

4. TCP 拥塞控制：回顾

在深入了解慢启动、拥塞避免和快速恢复的细节后，现在有必要退回来回顾一下全局。忽略一条连接开始时初始的慢启动阶段，假定丢包由 3 个冗余的 ACK 而不是超时指示，TCP 的拥塞控制是：每个 RTT 内 cwnd 线性（加性）增加 1MSS，然后出现 3 个冗余 ACK 事件时 cwnd 减半（乘性减）。因此，TCP 拥塞控制常常被称为**加性增、乘性减**（Additive-Increase，Multiplicative-Decrease，AIMD）拥塞控制方式。AIMD 拥塞控制引发了在图 3-53 中所示的“锯齿”行为，这也很好地图示了我们前面 TCP 检测带宽时的直觉，即 TCP 线性地增加它的拥塞窗口长度（因此增加其传输速率），直到出现 3 个冗余 ACK 事件。然后以 2 个因子来减少它的拥塞窗口长度，然后又开始了线性增长，探测是否还有另外的可用带宽。

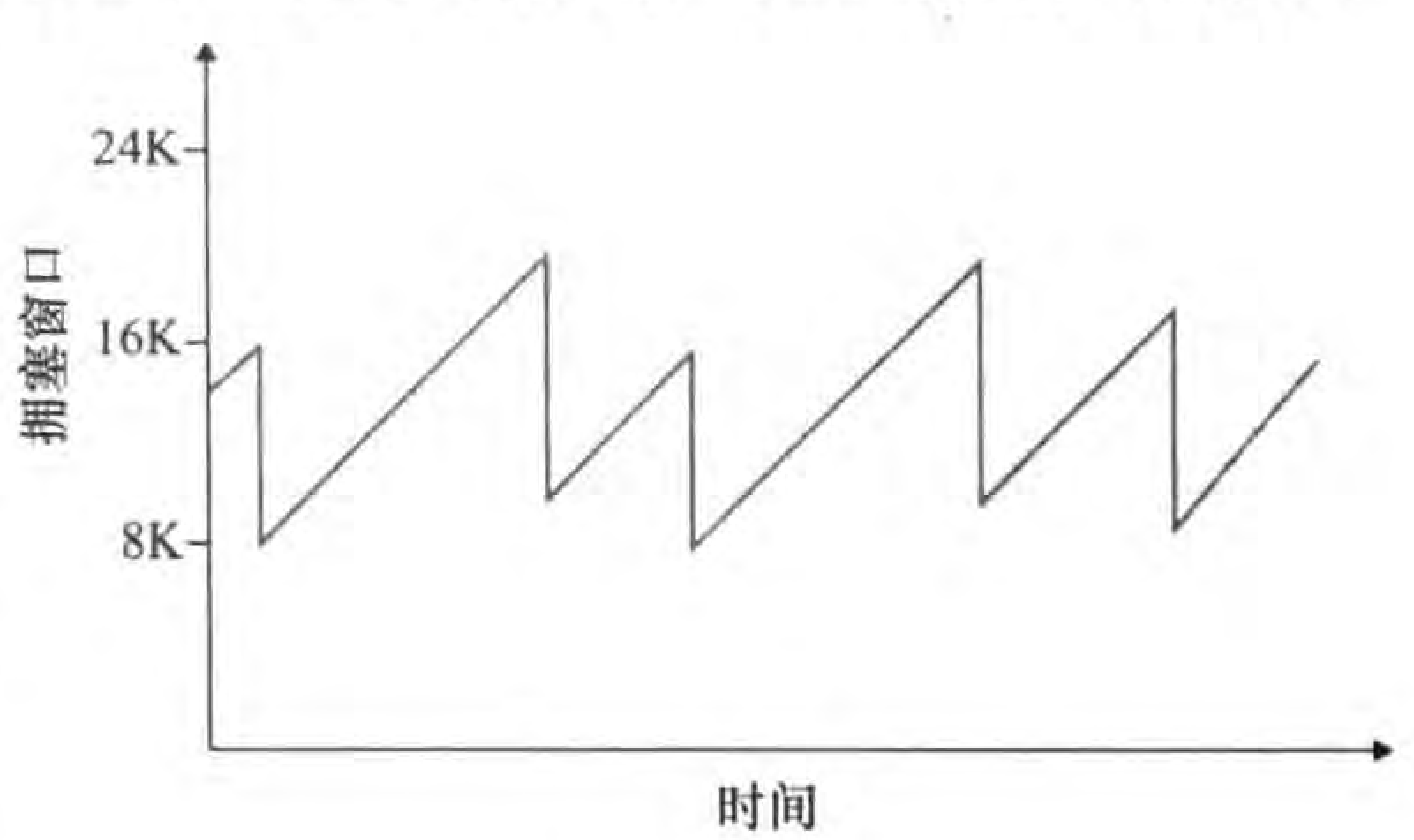


图 3-53 加性增、乘性减的拥塞控制

如前所述，许多 TCP 实现采用了 Reno 算法［Padhye 2001］。Reno 算法的许多变种已被提出［RFC 3782；RFC 2018］。TCP Vegas 算法［Brakmo 1995；Ahn 1995］试图在维持较好的吞吐量的同时避免拥塞。Vegas 的基本思想是：①在分组丢失发生之前，在源与目的地之间检测路由器中的拥塞；②当检测出快要发生的分组丢失时，线性地降低发送速率。快要发生的分组丢失是通过观察 RTT 来预测的。分组的 RTT 越长，路由器中的拥塞越严重。到 2015 年年底，TCP 的 Ubuntu Linux 实现默认提供了慢启动、拥塞避免、快速恢复、快速重传和 SACK，也提供了诸如 TCP Vegas 和 BIC［Xu 2004］等其他拥塞控制算法。对于许多特色 TCP 的综述参见［Afanasyev 2010］。

TCP AIMD 算法基于大量的工程见解和在运行网络中的拥塞控制经验而开发。在 TCP 研发后的十年，理论分析显示 TCP 的拥塞控制算法用做一种分布式异步优化算法，使得用户和网络性能的几个重要方面被同时优化 [Kelly 1998]。拥塞控制的丰富理论已经得到发展 [Srikant 2004]。

5. 对 TCP 吞吐量的宏观描述

给出 TCP 的锯齿状行为后，自然要考虑一个长存活期的 TCP 连接的平均吞吐量（即平均速率）可能是多少。在这个分析中，我们将忽略在超时事件后出现的慢启动阶段。（这些阶段通常非常短，因为发送方很快就以指数增长离开该阶段。）在一个特定的往返间隔内，TCP 发送数据的速率是拥塞窗口与当前 RTT 的函数。当窗口长度是 w 字节，且当前往返时间是 RTT 秒时，则 TCP 的发送速率大约是 w/RTT。于是，TCP 通过每经过 1 个 RTT 将 w 增加 1 个 MSS 探测出额外的带宽，直到一个丢包事件发生为止。当一个丢包事件发生时，用 W 表示 w 的值。假设在连接持续期间 RTT 和 W 几乎不变，那么 TCP 的传输速率在 $W/(2\times \text{RTT})$ 到 W/RTT 之间变化。

这些假设导出了 TCP 稳态行为的一个高度简化的宏观模型。当速率增长至 W/RTT 时，网络丢弃来自连接的分组；然后发送速率就会减半，进而每过一个 RTT 就发送速率增加 MSS/RTT，直到再次达到 W/RTT 为止。这一过程不断地自我重复。因为 TCP 吞吐量（即速率）在两个极值之间线性增长，所以我们有

$$\text{一条连接的平均吞吐量} = \frac{0.75 \times W}{\text{RTT}}$$

通过这个高度理想化的 TCP 稳态动态性模型，我们可以推出一个将连接的丢包率与可用带宽联系起来的有趣表达式 [Mahdavi 1997]。这个推导将在课后习题中概要给出。一个根据经验建立的并与测量数据一致的更复杂模型参见 [Padhye 2000]。

6. 经高带宽路径的 TCP

认识到下列事实是重要的：TCP 拥塞控制已经演化了多年并仍在继续演化。对当前 TCP 变量的总结和 TCP 演化的讨论，参见 [Floyd 2001；RFC 5681；Afanasyev 2010]。以往对因特网有益的东西（那时大量的 TCP 连接承载的是 SMTP、FTP 和 Telnet 流量），不一定对当今 HTTP 主宰的因特网或具有难以想象的服务的未来因特网还是有益的。

TCP 继续演化的需求能够通过考虑网格和云计算应用所需要的高速 TCP 连接加以阐述。例如，考虑一条具有 1500 字节报文段和 100ms RTT 的 TCP 连接，假定我们要通过这条连接以 10Gbps 速率发送数据。根据 [RFC 3649]，我们注意到使用上述 TCP 吞吐量公式，为了取得 10Gbps 吞吐量，平均拥塞窗口长度将需要是 83 333 个报文段。对如此大量的报文段，使我们相当关注这 83 333 个传输中的报文段也许会丢失。在丢失的情况下，将会出现什么情况呢？或者以另一种方式说，这些传输的报文段能以何种比例丢失，使得在图 3-52 中列出的 TCP 拥塞控制算法仍能取得所希望的 10Gbps 速率？在本章的课后习题中，要求读者推导出一条 TCP 连接的吞吐量公式，该公式作为丢包率（L）、往返时间（RTT）和最大报文段长度（MSS）的函数：

$$\text{一条连接的平均吞吐量} = \frac{1.22 \times \text{MSS}}{\text{RTT}\sqrt{L}}$$

使用该公式，我们能够看到，为了取得 10Gbps 的吞吐量，今天的 TCP 拥塞控制算法

仅能容忍 2×10^{-10} 的报文段丢失概率（或等价地说，对每 5 000 000 000 个报文段有一个丢包），这是一个非常低的值。这种观察导致许多研究人员为这种高速环境特别设计新版 TCP，对这些努力的讨论参见［Jin 2004；Kelly 2003；Ha 2008；RFC 7323］。

3.7.1 公平性

考虑 K 条 TCP 连接，每条都有不同的端到端路径，但是都经过一段传输速率为 R bps 的瓶颈链路。（所谓*瓶颈链路*，是指对于每条连接，沿着该连接路径上的所有其他段链路都不拥塞，而且与该瓶颈链路的传输容量相比，它们都有充足的传输容量。）假设每条连接都在传输一个大文件，而且无 UDP 流量通过该段瓶颈链路。如果每条连接的平均传输速率接近 R/K，即每条连接都得到相同份额的链路带宽，则认为该拥塞控制机制是*公平的*。

TCP 的 AIMD 算法公平吗？尤其是假定可在不同时间启动并因此在某个给定的时间点可能具有不同的窗口长度情况下，对这些不同的 TCP 连接还是公平的吗？TCP 趋于在竞争的多条 TCP 连接之间提供对一段瓶颈链路带宽的平等分享，其理由［Chiu 1989］给出了一个极好的、直观的解释。

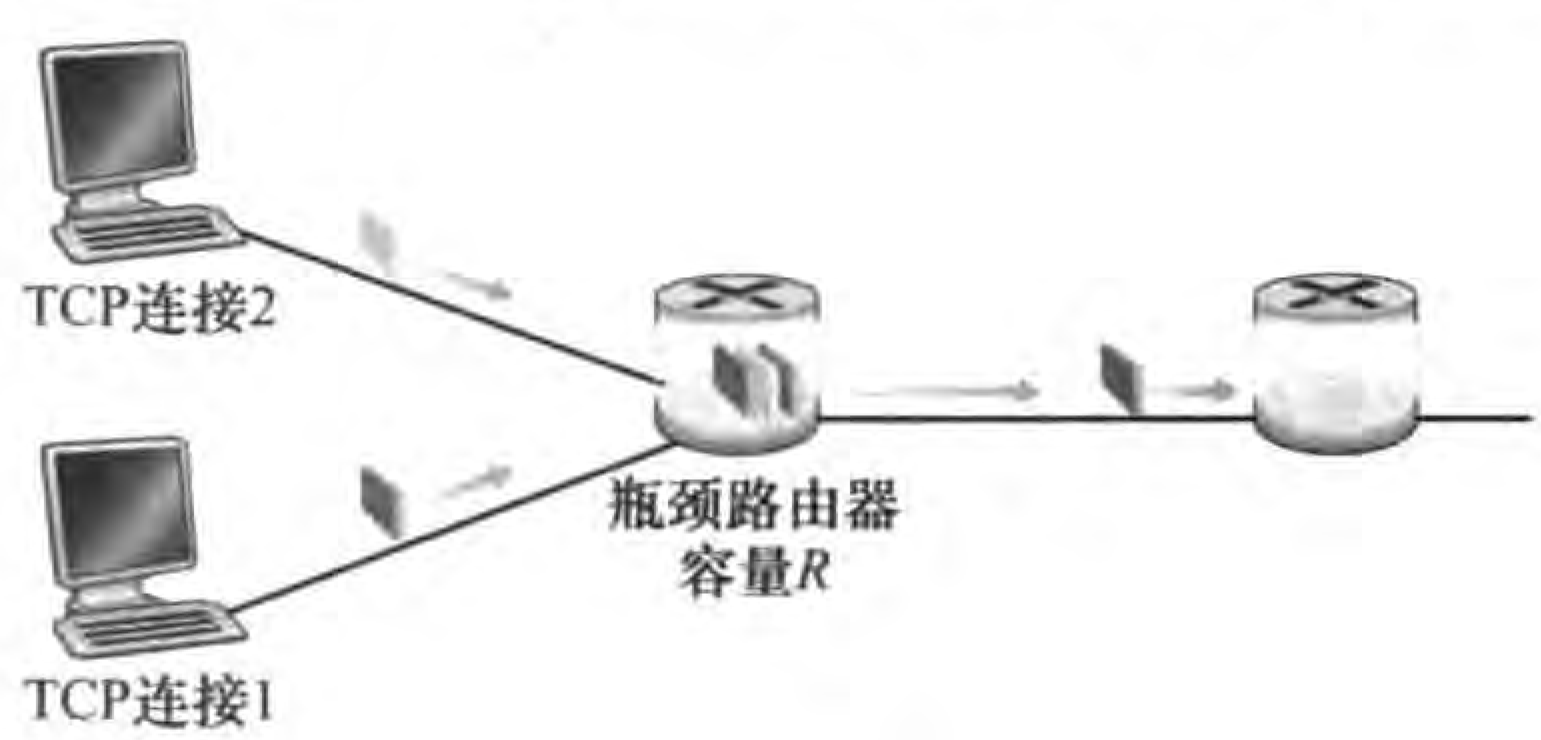


图 3-54 两条 TCP 连接共享同一条瓶颈链路

我们考虑有两条 TCP 连接共享一段传输速率为 R 的链路的简单例子，如图 3-54 中所示。我们将假设这两条连接有相同的 MSS 和 RTT（这样如果它们有相同的拥塞窗口长度，就会有相同的吞吐量），它们有大量的数据要发送，且没有其他 TCP 连接或 UDP 数据报穿越该段共享链路。我们还将忽略 TCP 的慢启动阶段，并假设 TCP 连接一直按 CA 模式（AIMD）运行。

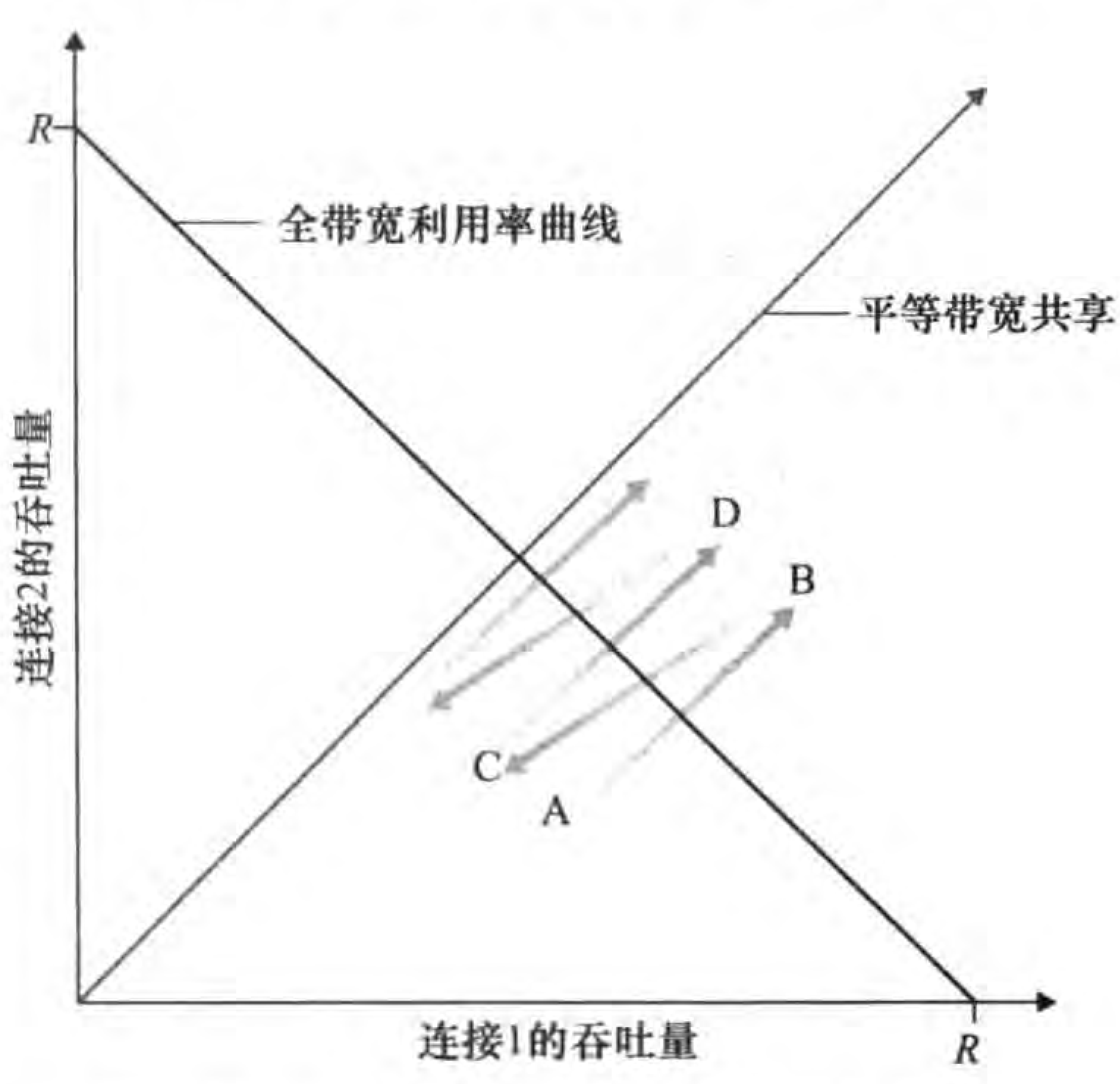


图 3-55 TCP 连接 1 和连接 2 实现的吞吐量

图 3-55 描绘了两条 TCP 连接实现的吞吐量情况。如果 TCP 要在这两条 TCP 连接之间平等地共享链路带宽，那么实现的吞吐量曲线应当是从原点沿 45°方向的箭头向外辐射（平等带宽共享）。理想情况是，两个吞吐量的和应等于 R。（当然，每条连接得到相同但容量为 0 的共享链路容量并非我们所期望的情况！）所以我们的目标应该是使取得的吞吐量落在图 3-55 中平等带宽共享曲线与全带宽利用曲线的交叉点附近的某处。

假定 TCP 窗口长度是这样的，即在某给定时刻，连接 1 和连接 2 实现了由图 3-55 中 A 点所指明的吞吐量。因为这两条连接共同消耗的链路带宽量小于 R，所以无丢包事件发生，根据 TCP 的拥塞避免算法的结果，这两条连接每过一个 RTT 都要将其窗口增加 1 个 MSS。因此，这两条连接的总吞吐量就会从 A 点开始沿 45°线前行（两条连接都有相同的增长）。最终，这两条连接共同消耗的带宽将超过 R，最终将发生分组丢失。假设连接 1

和连接2实现B点指明的吞吐量时，它们都经历了分组丢失。连接1和连接2于是就按二分之一减小其窗口。所产生的结果实现了C点指明的吞吐量，它正好位于始于B点止于原点的一个向量的中间。因为在C点，共同消耗的带宽小于 R，所以这两条连接再次沿着始于C点的45°线增加其吞吐量。最终，再次发生丢包事件，如在D点，这两条连接再次将其窗口长度减半，如此等等。你应当搞清楚这两条连接实现的带宽最终将沿着平等带宽共享曲线在波动。还应该搞清楚无论这两条连接位于二维空间的何处，它们最终都会收敛到该状态！虽然此时我们做了许多理想化的假设，但是它仍然能对解释为什么TCP会导致在多条连接之间的平等共享带宽这个问题提供一个直观的感觉。

在理想化情形中，我们假设仅有TCP连接穿过瓶颈链路、所有的连接具有相同的RTT值，且对于一个主机-目的地对而言只有一条TCP连接与之相关联。实践中，这些条件通常是得不到满足的，客户-服务器应用因此能获得非常不平等的链路带宽份额。特别是，已经表明当多条连接共享一个共同的瓶颈链路时，那些具有较小RTT的连接能够在链路空闲时更快地抢到可用带宽（即较快地打开其拥塞窗口），因而将比那些具有较大RTT的连接享用更高的吞吐量［Laksman 1997］。

1. 公平性和UDP

我们刚才已经看到，TCP拥塞控制是如何通过拥塞窗口机制来调节一个应用程序的传输速率的。许多多媒体应用如因特网电话和视频会议，经常就因为这种特定原因而不在TCP上运行，因为它们不想其传输速率被扼制，即使在网络非常拥塞的情况下。相反，这些应用宁可在UDP上运行，UDP是没有内置的拥塞控制的。当运行在UDP上时，这些应用能够以恒定的速率将其音频和视频数据注入网络之中并且偶尔会丢失分组，而不愿在拥塞时将其发送速率降至“公平”级别并且不丢失任何分组。从TCP的观点来看，运行在UDP上的多媒体应用是不公平的，因为它们不与其他连接合作，也不适时地调整其传输速率。因为TCP拥塞控制在面临拥塞增加（丢包）时，将降低其传输速率，而UDP源则不必这样做，UDP源有可能压制TCP流量。当今的一个主要研究领域就是开发一种因特网中的拥塞控制机制，用于阻止UDP流量不断压制直至中断因特网吞吐量的情况［Floyd 1999；Floyd 2000；Kohler 2006；RFC 4340］。

2. 公平性和并行TCP连接

即使我们能够迫使UDP流量具有公平的行为，但公平性问题仍然没有完全解决。这是因为我们没有什么办法阻止基于TCP的应用使用多个并行连接。例如，Web浏览器通常使用多个并行TCP连接来传送一个Web页中的多个对象。（多条连接的确切数目可以在多数浏览器中进行配置。）当一个应用使用多条并行连接时，它占用了一条拥塞链路中较大比例的带宽。举例来说，考虑一段速率为 R 且支持9个在线客户-服务器应用的链路，每个应用使用一条TCP连接。如果一个新的应用加入进来，也使用一条TCP连接，则每个应用得到差不多相同的传输速率 $R/10$。但是如果这个新的应用这次使用了11个并行TCP连接，则这个新应用就不公平地分到超过 $R/2$ 的带宽。Web流量在因特网中是非常普遍的，所以多条并行连接并非不常见。

3.7.2 明确拥塞通告：网络辅助拥塞控制

自20世纪80年代后期慢启动和拥塞避免开始标准化以来［RFC 1122］，TCP已经实现了端到端拥塞控制的形式，我们在3.7.1节中对此进行了学习：一个TCP发送方不会收

到来自网络层的明确拥塞指示，而是通过观察分组丢失来推断拥塞。最近，对于 IP 和 TCP 的扩展方案［RFC 3168］已经提出并已经实现和部署，该方案允许网络明确向 TCP 发送方和接收方发出拥塞信号。这种形式的网络辅助拥塞控制称为**明确拥塞通告**（Explicit Congestion Notification，ECN）。如图 3-56 所示，涉及了 TCP 和 IP 协议。

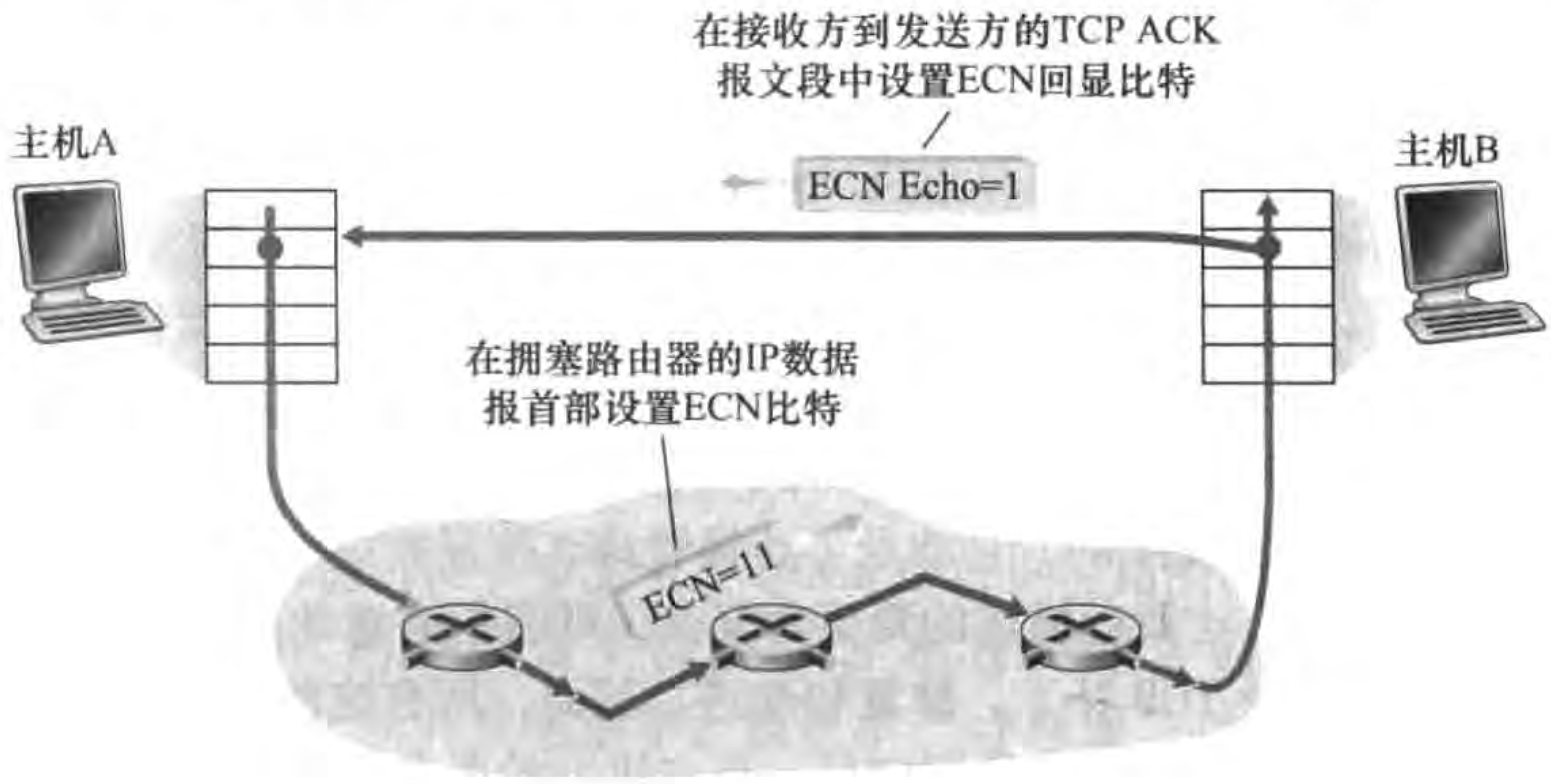


图 3-56 明确拥塞通告：网络辅助拥塞控制

在网络层，IP 数据报首部的服务类型字段中的两个比特（总的说来，有四种可能的值）被用于 ECN。路由器所使用的一种 ECN 比特设置指示该路由器正在历经拥塞。该拥塞指示则由被标记的 IP 数据报所携带，送给目的主机，再由目的主机通知发送主机，如图 3-56 所示。RFC 3168 没有提供路由器拥塞时的定义；该判断是由路由器厂商所做的配置选择，并且由网络操作员决定。然而，RFC 3168 推荐仅当拥塞持续不断存在时才设置 ECN 比特。发送主机所使用的另一种 ECN 比特设置通知路由器发送方和接收方是 ECN 使能的，因此能够对于 ECN 指示的网络拥塞采取行动。

如图 3-56 所示，当接收主机中的 TCP 通过一个接收到的数据报收到了一个 ECN 拥塞指示时，接收主机中的 TCP 通过在接收方到发送方的 TCP ACK 报文段中设置 ECE（明确拥塞通告回显）比特（参见图 3-29），通知发送主机中的 TCP 收到拥塞指示。接下来，TCP 发送方通过减半拥塞窗口对一个具有 ECE 拥塞指示的 ACK 做出反应，就像它对丢失报文段使用快速重传做出反应一样，并且在下一个传输的 TCP 发送方到接收方的报文段首部中对 CWR（拥塞窗口缩减）比特进行设置。

除了 TCP 以外的其他运输层协议也可以利用网络层发送 ECN 信号。数据报拥塞控制协议（Datagram Congestion Control Protocol，DCCP）［RFC 4340］提供了一种低开销、控制拥塞的类似 UDP 不可靠服务，该协议利用了 ECN。DCTCP（数据中心 TCP）［Alizadeh 2010］是一种专门为数据中心网络设计的 TCP 版本，也利用了 ECN。

3.8 小结

本章我们首先学习了运输层协议能够向网络应用程序提供的服务。在一个极端，运输层协议非常简单，并向应用程序不提供不必要的服务，而仅向通信进程提供多路复用/分解的功能。因特网中的 UDP 协议就是这样一种不提供不必要服务的运输层协议。在另一个极端，运输层协议能够向应用程序提供各种各样的保证，例如数据的可靠交付、时延保证和带宽保证。无论如何，运输层协议能够提供的服务经常受下面网络层协议服务模型的

限制。如果网络层协议不能向运输层报文段提供时延或带宽保证，那么运输层协议就不能向进程间发送的报文提供时延或带宽保证。

在3.4节中，我们学习了运输层协议能够提供可靠数据传输，即使下面的网络层是不可靠的。我们看到了提供可靠的数据传送会遇到许多微妙的问题，但都可以通过精心地结合确认、定时器、重传以及序号机制来完成任务。

尽管在本章中我们包含了可靠数据传送，但是我们应该理解在链路层、网络层、运输层或应用层协议中都可以提供可靠数据传送。该协议栈中上面4层的任意一层都可以实现确认、定时器、重传以及序号，能够向其上层提供可靠数据传送。事实上，在过去数年中，工程师以及计算机科学家们已经独立地设计并实现了提供可靠数据传送的链路层、网络层、运输层以及应用层协议（虽然这些协议中的许多已经销声匿迹了）。

在3.5节中，我们详细地研究了TCP协议，它是因特网中面向连接和可靠的运输层协议。我们知道TCP是非常复杂的，它涉及了连接管理、流量控制、往返时间估计以及可靠数据传送。事实上，TCP比我们描述的要更为复杂，即我们有意地避而不谈在各种TCP实现版本中广泛实现的各种TCP补丁、修复和改进。然而，所有这些复杂性都对网络层应用隐藏了起来。如果某主机上的客户希望向另一台主机上的服务器可靠地发送数据，它只需要打开对该服务器的一个TCP套接字，然后将数据注入该套接字。客户-服务器应用程序则乐于对TCP的复杂性视而不见。

在3.6节中，我们从广泛的角度研究了拥塞控制，在3.7节中我们阐述了TCP是如何实现拥塞控制的。我们知道了拥塞控制对于网络良好运行是必不可少的。没有拥塞控制，网络很容易出现死锁，使得端到端之间很少或没有数据能被传输。在3.7节中我们学习了TCP实现的一种端到端拥塞控制机制，即当TCP连接的路径上判断不拥塞时，其传输速率就加性增；当出现丢包时，传输速率就乘性减。这种机制也致力于做到每一个通过拥塞链路的TCP连接能平等地共享该链路带宽。我们也深入探讨了TCP连接建立和慢启动对时延的影响。我们观察到在许多重要场合，连接建立和慢启动会对端到端时延产生严重影响。我们再次强调，尽管TCP在这几年一直在发展，但它仍然是一个值得深入研究的领域，并且在未来的几年中还可能持续演化。

在本章中我们对特定因特网运输协议的讨论集中在UDP和TCP上，它们是因特网运输层的两匹“驮马”。然而，对这两个协议的二十多年的经验已经使人们认识到，这两个协议都不是完美无缺的。研究人员因此在忙于研制其他的运输层协议，其中的几种现在已经成为IETF建议的标准。

数据报拥塞控制协议（Datagram Congestion Control Protocol，DCCP）[RFC 4340] 提供了一种低开销、面向报文、类似于UDP的不可靠服务，但是具有应用程序可选择的拥塞控制形式，该机制与TCP相兼容。如果某应用程序需要可靠的或半可靠的数据传送，则这将在应用程序自身中执行（也许使用我们已经在3.4节中学过的机制）。DCCP被设想用于诸如流媒体（参见第9章）等应用程序中，DCCP能够利用数据交付的预定时间和可靠性之间的折中，但是要对网络拥塞做出响应。

在谷歌的Chromium浏览器中实现了QUIC（Quick UDP Internet Connections）协议[Iyengar 2016]，该协议通过重传以及差错检测、快速连接建立和基于速率的拥塞控制算法提供可靠性，而基于速率的拥塞控制算法是以TCP友好特性为目标，这些机制都是在UDP之上作为应用层协议实现的。2015年年初，谷歌报告从Chrome浏览器到谷歌服务器的大约一半请求运行在QUIC之上。

DCTCP（数据中心 TCP）[Alizadeh 2010] 是一种专门为数据中心网络设计的 TCP 版本，使用 ECN 以更好地支持短流和长流的混合流，这种混合流代表了数据中心负载的特征。

流控制传输协议（Stream Control Transmission Protocol，SCTP）[RFC 4960，RFC 3286] 是一种可靠的、面向报文的协议，该协议允许几个不同的应用层次的“流”复用到单个 SCTP 连接上（一种称之为“多流”的方法）。从可靠性的角度看，在该连接中的不同流被分别处理，因此在一条流中的分组丢失不会影响其他流中数据的交付。当一台主机与两个或更多个网络连接时，SCTP 也允许数据经两条出路径传输，还具有失序数据的选项交付和一些其他特色。SCTP 的流控制和拥塞控制算法基本上与 TCP 中的相同。

TCP 友好速率控制（TCP-Friendly Rate Control，TFRC）协议 [RFC 5348] 是一种拥塞控制协议而不是一种功能齐全的运输层协议。它定义了一种拥塞控制机制，该机制能被用于诸如 DCCP 等其他运输协议（事实上在 DCCP 中可供使用的两种应用程序可选的协议之一就是 TFRC）。TFRC 的目标是平滑在 TCP 拥塞控制中的“锯齿”行为（参见图 3-53），同时维护一种长期的发送速率，该速率“合理地”接近 TCP 的速率。使用比 TCP 更为平滑的发送速率，TFRC 非常适合诸如 IP 电话或流媒体等多媒体应用，这种平滑的速率对于这些应用是重要的。TFRC 是一种“基于方程”的协议，这些协议使用测得的丢包率作为方程的输入 [Padhye 2000]，即使用方程估计一个 TCP 会话在该丢包率下 TCP 的吞吐量将是多大。该速率则被取为 TFRC 的目标发送速率。

唯有未来才能告诉我们 DCCP、SCTP、QUIC 或 TFRC 是否能得到广泛实施。虽然这些协议明确地提供了超过 TCP 和 UDP 的强化能力，但是多年来已经证明了 TCP 和 UDP 自身是“足够好”的。是否“更好”将胜出“足够好”，这将取决于技术、社会和商业考虑的复杂组合。

在第 1 章中，我们讲到计算机网络能被划分成“网络边缘”和“网络核心”。网络边缘包含了在端系统中发生的所有事情。既然已经覆盖了应用层和运输层，我们关于网络边缘的讨论也就结束了。接下来是探寻网络核心的时候了！我们的旅程从下一章开始，下一章将学习网络层，并且将在第 6 章继续学习链路层。

课后习题和问题

复习题

3.1 ~ 3.3 节

R1. 假定网络层提供了下列服务。在源主机中的网络层接受最大长度 1200 字节和来自运输层的目的主机地址的报文段。网络层则保证将该报文段交付给位于目的主机的运输层。假定在目的主机上能够运行许多网络应用进程。

a. 设计可能最简单的运输层协议，该协议将使应用程序数据到达位于目的主机的所希望的进程。假设在目的主机中的操作系统已经为每个运行的应用进程分配了一个 4 字节的端口号。

b. 修改这个协议，使它向目的进程提供一个的“返回地址”。

c. 在你的协议中，该运输层在计算机网络的核心中“必须做任何事”吗？

R2. 考虑有一个星球，每个人都属于某个六口之家，每个家庭都住在自己的房子里，每个房子都一个唯一的地址，并且某给定家庭中的每个人有一个独特的名字。假定该星球有一个从源家庭到目的家庭交付信件的邮政服务。该邮件服务要求：①在一个信封中有一封信；②在信封上清楚地写上目的家

庭的地址（并且没有别的东西）。假设每个家庭有一名家庭成员代表为家庭中的其他成员收集和分发信件。这些信没有必要提供任何有关信的接收者的指示。

a. 使用对上面复习题R1的解决方案作为启发，描述家庭成员代表能够使用的协议，以从发送家庭成员向接收家庭成员交付信件。

b. 在你的协议中，该邮政服务必须打开信封并检查信件内容才能提供它的服务吗？

R3. 考虑在主机A和主机B之间有一条TCP连接。假设从主机A传送到主机B的TCP报文段具有源端口号x和目的端口号y。对于从主机B传送到主机A的报文段，源端口号和目的端口号分别是多少？

R4. 描述应用程序开发者为什么可能选择在UDP上运行应用程序而不是在TCP上运行的原因。

R5. 在今天的因特网中，为什么语音和图像流量常常是经过TCP而不是经UDP发送。（提示：我们寻找的答案与TCP的拥塞控制机制没有关系。）

R6. 当某应用程序运行在UDP上时，该应用程序可能得到可靠数据传输吗？如果能，如何实现？

R7. 假定在主机C上的一个进程有一个具有端口号6789的UDP套接字。假定主机A和主机B都用目的端口号6789向主机C发送一个UDP报文段。这两台主机的这些报文段在主机C都被描述为相同的套接字吗？如果是这样的话，在主机C的该进程将怎样知道源于两台不同主机的这两个报文段？

R8. 假定在主机C端口80上运行的一个Web服务器。假定这个Web服务器使用持续连接，并且正在接收来自两台不同主机A和B的请求。被发送的所有请求都通过位于主机C的相同套接字吗？如果它们通过不同的套接字传递，这两个套接字都具有端口80吗？讨论和解释之。

3.4节

R9. 在我们的rdt协议中，为什么需要引入序号？

R10. 在我们的rdt协议中，为什么需要引入定时器？

R11. 假定发送方和接收方之间的往返时延是固定的并且为发送方所知。假设分组能够丢失的话，在协议rdt3.0中，一个定时器仍是必需的吗？试解释之。

R12. 在配套网站上使用Go-Back-N（回退N步）Java小程序。

a. 让源发送5个分组，在这5个分组的任何一个到达目的地之前暂停该动画。然后毁掉第一个分组并继续该动画。试描述发生的情况。

b. 重复该实验，只是现在让第一个分组到达目的地并毁掉第一个确认。再次描述发生的情况。

c. 最后，尝试发送6个分组。发生了什么情况？

R13. 重复复习题R12，但是现在使用Selective Repeat（选择重传）Java小程序。选择重传和回退N步有什么不同？

3.5节

R14. 是非判断题：

a. 主机A经过一条TCP连接向主机B发送一个大文件。假设主机B没有数据发往主机A。因为主机B不能随数据捎带确认，所以主机B将不向主机A发送确认。

b. 在连接的整个过程中，TCP的rwnd的长度决不会变化。

c. 假设主机A通过一条TCP连接向主机B发送一个大文件。主机A发送但未被确认的字节数不会超过接收缓存的大小。

d. 假设主机A通过一条TCP连接向主机B发送一个大文件。如果对于这条连接的一个报文段的序号为m，则对于后继报文段的序号将必然是$m+1$。

e. TCP报文段在它的首部中有一个rwnd字段。

f. 假定在一条TCP连接中最后的SampleRTT等于1秒，那么对于该连接的TimeoutInterval的当前值必定大于等于1秒。

g. 假设主机A通过一条TCP连接向主机B发送一个序号为38的4个字节的报文段。在这个相同的报文段中，确认号必定是42。

R15. 假设主机A通过一条TCP连接向主机B发送两个紧接着的TCP报文段。第一个报文段的序号为90，第二个报文段序号为110。

a. 第一个报文段中有多少数据?

b. 假设第一个报文段丢失而第二个报文段到达主机 B。那么在主机 B 发往主机 A 的确认报文中，确认号应该是多少?

R16. 考虑在 3.5 节中讨论的 Telnet 的例子。在用户键入字符 C 数秒之后，用户又键入字符 R。那么在用户键入字符 R 之后，总共发送了多少个报文段，这些报文段中的序号和确认字段应该填入什么?

3.7 节

R17. 假设两条 TCP 连接存在于一个带宽为 *R* bps 的瓶颈链路上。它们都要发送一个很大的文件（以相同方向经过瓶颈链路)，并且两者是同时开始发送文件。那么 TCP 将为每条连接分配什么样的传输速率?

R18. 是非判断题。考虑 TCP 的拥塞控制。当发送方定时器超时时，其 ssthresh 的值将被设置为原来值的一半。

R19. 在 3.7 节的“TCP 分岔”讨论中，对于 TCP 分岔的响应时间，断言大约是 $4 * RTT_{FE} + RTT_{BE}$ + 处理时间。评价该断言。

习题

P1. 假设客户 A 向服务器 S 发起一个 Telnet 会话。与此同时，客户 B 也向服务器 S 发起一个 Telnet 会话。给出下面报文段的源端口号和目的端口号：

a. 从 A 向 S 发送的报文段。

b. 从 B 向 S 发送的报文段。

c. 从 S 向 A 发送的报文段。

d. 从 S 向 B 发送的报文段。

e. 如果 A 和 B 是不同的主机，那么从 A 向 S 发送的报文段的源端口号是否可能与从 B 向 S 发送的报文段的源端口号相同?

f. 如果它们是同一台主机，情况会怎么样?

P2. 考虑图 3-5。从服务器返回客户进程的报文流中的源端口号和目的端口号是多少? 在承载运输层报文段的网络层数据报中，IP 地址是多少?

P3. UDP 和 TCP 使用反码来计算它们的检验和。假设你有下面 3 个 8 比特字节：01010011，01100110，01110100。这些 8 比特字节和的反码是多少?（注意到尽管 UDP 和 TCP 使用 16 比特的字来计算检验和，但对于这个问题，你应该考虑 8 比特和。）写出所有工作过程。UDP 为什么要用该和的反码，即为什么不直接使用该和呢? 使用该反码方案，接收方如何检测出差错? 1 比特的差错将可能检测不出来吗? 2 比特的差错呢?

P4. a. 假定你有下列 2 个字节：01011100 和 01100101。这 2 个字节之和的反码是什么?

b. 假定你有下列 2 个字节：11011010 和 01100101。这 2 个字节之和的反码是什么?

c. 对于（a）中的字节，给出一个例子，使得这 2 个字节中的每一个都在一个比特反转时，其反码不会改变。

P5. 假定某 UDP 接收方对接收到的 UDP 报文段计算因特网检验和，并发现它与承载在检验和字段中的值相匹配。该接收方能够绝对确信没有出现过比特差错吗? 试解释之。

P6. 考虑我们改正协议 rdt2.1 的动机。试说明如图 3-57 所示的接收方与如图 3-11 所示的发送方运行时，接收方可能会引起发送方和接收方进入死锁状态，即双方都在等待不可能发生的事件。

P7. 在 rdt3.0 协议中，从接收方向发送方流动的 ACK 分组没有序号（尽管它们具有 ACK 字段，该字段包括了它们正在确认的分组的序号)。为什么这些 ACK 分组不需要序号呢?

P8. 画出协议 rdt3.0 中接收方的 FSM。

P9. 当数据分组和确认分组发生篡改时，给出 rdt3.0 协议运行的轨迹。你画的轨迹应当类似于图 3-16 中所用的图。

P10. 考虑一个能够丢失分组但其最大时延已知的信道。修改协议 rdt2.1，以包括发送方超时和重传机制。

非正式地论证：为什么你的协议能够通过该信道正确通信？

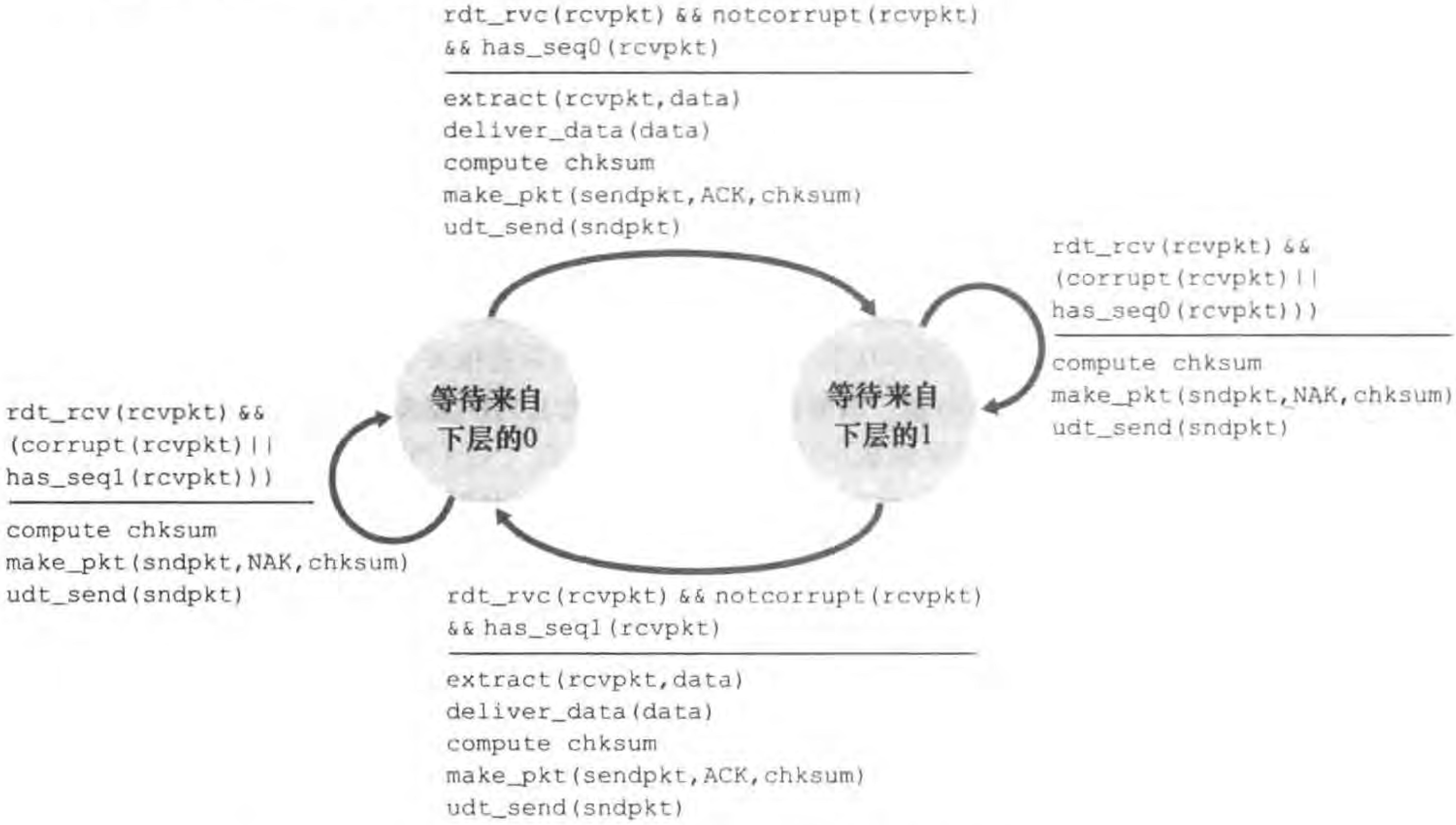


图 3-57 协议 rdt2.1 的一个不正确的接收方

P11. 考虑在图 3-14 中的 rdt2.2 接收方，在状态“等待来自下层的 0”和状态“等待来自下层的 1”中的自转换（即从某状态转换回自身）中生成一个新分组：sndpk = make_ pkt（ACK，1，checksum）和 sndpk = make_ pkt（ACK，0，checksum）。如果这个动作从状态“等待来自下层的 1”中的自转换中删除，该协议将正确工作吗？评估你的答案。在状态“等待来自下层的 0”中的自转换中删除这个事件将会怎样？[提示：在后一种情况下，考虑如果第一个发送方到接收方的分组损坏的话，将会发生什么情况？]

P12. rdt3.0 协议的发送方直接忽略（即不采取任何动作）接收到的所有出现差错和确认分组的确认号（acknum）字段中的值有差错的分组。假设在这种情况下，rdt3.0 只是重传当前的数据分组，该协议是否还能正常运行？（提示：考虑在下列情况下会发生什么情况：仅有一个比特差错时；报文没有丢失但能出现定时器过早超时。考虑到当 n 趋于无穷时，第 n 个分组将被发送多少次。）

P13. 考虑 rdt3.0 协议。如果发送方和接收方的网络连接能够对报文重排序（即在发送方和接收方之间的媒体上传播的两个报文段能重新排序），那么比特交替协议将不能正确工作（确信你清楚地理解这时它不能正确工作的原因），试画图说明之。画图时把发送方放在左边，接收方放在右边，使时间轴朝下，标出交换的数据报文（D）和确认报文（A）。要标明与任何数据和确认报文段相关的序号。

P14. 考虑一种仅使用否定确认的可靠数据传输协议。假定发送方只是偶尔发送数据。只用 NAK 的协议是否会比使用 ACK 的协议更好？为什么？现在我们假设发送方要发送大量的数据，并且该端到端连接很少丢包。在第二种情况下，只用 NAK 的协议是否会比使用 ACK 的协议更好？为什么？

P15. 考虑显示在图 3-17 中的网络跨越国家的例子。窗口长度设置成多少时，才能使该信道的利用率超过 90%？假设分组的长度为 1500 字节（包括首部字段和数据）。

P16. 假设某应用使用 rdt3.0 作为其运输层协议。因为停等协议具有非常低的信道利用率（显示在网络跨越国家的例子中），该应用程序的设计者让接收方持续回送许多（大于 2）交替的 ACK 0 和 ACK 1，即使对应的数据未到达接收方。这个应用程序设计将能增加信道利用率吗？为什么？该方法存在某种潜在的问题吗？试解释之。

P17. 考虑两个网络实体 A 和 B，它们由一条完善的双向信道所连接（即任何发送的报文将正确地收到；

信道将不会损坏、丢失或重排序分组）。A 和 B 将以交互的方式彼此交付报文：首先，A 必须向 B 交付一个报文，B 然后必须向 A 交付一个报文，接下来 A 必须向 B 交付一个报文，等等。如果一个实体处于它不试图向另一侧交付报文的状态，将存在一个来自上层的类似于 rdt_send(data) 调用的事件，它试图向下传送数据以向另一侧传输，来自上层的该调用能够直接忽略对于 rdt_unable_to_send(data) 调用，这通知较高层当前不能够发送数据。[注意：做出这种简化的假设，使你不必担心缓存数据。]

对该协议画出 FSM 说明（一个 FSM 用于 A，一个 FSM 用于 B）。注意你不必担心这里的可靠性机制，该问题的要点在于创建反映这两个实体的同步行为的 FSM 说明。应当使用与图 3-9 中协议 rdt1.0 有相同含义的下列事件和动作：rdt_send(data)，packet = make_pkt(data)，udt__send(data)，rdt_rcv(packet)，extract(packet,data)，deliver_data(data)。保证你的协议反映了 A 和 B 之间发送的严格交替。还要保证在你的 FSM 描述中指出 A 和 B 的初始状态。

P18. 在 3.4.4 节我们学习的一般性 SR 协议中，只要报文可用（如果报文在窗口中），发送方就会不等待确认而传输报文。假设现在我们要求一个 SR 协议，一次发出一对报文，而且只有在知道第一对报文中的两个报文都正确到达后才发送第二对报文。

假设该信道中可能会丢失报文，但报文不会发生损坏和失序。试为报文的单向可靠传输而设计一个差错控制协议。画出发送方和接收方的 FSM 描述。描述在发送方和接收方之间两个方向发送的报文格式。如果你使用了不同于 3.4 节（例如 udt_send()、start_timer()、rdt_rcv()等）中的任何其他过程调用，详细地阐述这些动作。举例说明（用发送方和接收方的时序踪迹图）你的协议是如何恢复报文丢失的。

P19. 考虑一种情况，主机 A 想同时向主机 B 和主机 C 发送分组。A 与 B 和 C 是经过广播信道连接的，即由 A 发送的分组通过该信道传送到 B 和 C。假设连接 A、B 和 C 的这个广播信道具有独立的报文丢失和损坏特性（例如，从 A 发出的报文可能被 B 正确接收，但没有被 C 正确接收）。设计一个类似于停等协议的差错控制协议，用于从 A 可靠地传输分组到 B 和 C。该协议使得 A 直到得知 B 和 C 已经正确接收到当前报文，才获取上层交付的新数据。给出 A 和 C 的 FSM 描述。（提示：B 的 FSM 大体上应当与 C 的相同。）同时，给出所使用的报文格式的描述。

P20. 考虑一种主机 A 和主机 B 要向主机 C 发送报文的情况。主机 A 和 C 通过一条报文能够丢失和损坏（但不重排序）的信道相连接。主机 B 和 C 由另一条（与连接 A 和 C 的信道独立）具有相同性质的信道连接。在主机 C 上的运输层，在向上层交付来自主机 A 和 B 的报文时应当交替进行（即它应当首先交付来自 A 的分组中的数据，然后是来自 B 的分组中的数据，等等）。设计一个类似于停等协议的差错控制协议，以可靠地向 C 传输来自 A 和 B 的分组，同时以前面描述的方式在 C 处交替地交付。给出 A 和 C 的 FSM 描述。（提示：B 的 FSM 大体上应当与 A 的相同。）同时，给出所使用的报文格式的描述。

P21. 假定我们有两个网络实体 A 和 B。B 有一些数据报文要通过下列规则传给 A。当 A 从其上层得到一个请求，就从 B 获取下一个数据（D）报文。A 必须通过 A—B 信道向 B 发送一个请求（R）报文。仅当 B 收到一个 R 报文后，它才会通过 B—A 信道向 A 发送一个数据（D）报文。A 应当准确地将每份 D 报文的副本交付给上层。R 报文可能会在 A—B 信道中丢失（但不会损坏）；D 报文一旦发出总是能够正确交付。两个信道的时延未知且是变化的。

设计一个协议（给出 FSM 描述），它能够综合适当的机制，以补偿会丢包的 A—B 信道，并且实现在 A 实体中向上层传递报文。只采用绝对必要的机制。

P22. 考虑一个 GBN 协议，其发送方窗口为 4，序号范围为 1024。假设在时刻 t，接收方期待的下一个有序分组的序号是 k。假设媒体不会对报文重新排序。回答以下问题：

a. 在 t 时刻，发送方窗口内的报文序号可能是多少？论证你的回答。

b. 在 t 时刻，在当前传播回发送方的所有可能报文中，ACK 字段的所有可能值是多少？论证你的回答。

P23. 考虑 GBN 协议和 SR 协议。假设序号空间的长度为 k，那么为了避免出现图 3-27 中的问题，对于这

两种协议中的每一种，允许的发送方窗口最大为多少？

P24. 对下面的问题判断是非，并简要地证实你的回答：

a. 对于 SR 协议，发送方可能会收到落在其当前窗口之外的分组的 ACK。

b. 对于 GBN 协议，发送方可能会收到落在其当前窗口之外的分组的 ACK。

c. 当发送方和接收方窗口长度都为 1 时，比特交替协议与 SR 协议相同。

d. 当发送方和接收方窗口长度都为 1 时，比特交替协议与 GBN 协议相同。

P25. 我们曾经说过，应用程序可能选择 UDP 作为运输协议，因为 UDP 提供了（比 TCP）更好的应用层控制，以决定在报文段中发送什么数据和发送时机。

a. 应用程序为什么对在报文段中发送什么数据有更多的控制？

b. 应用程序为什么对何时发送报文段有更多的控制？

P26. 考虑从主机 A 向主机 B 传输 L 字节的大文件，假设 MSS 为 536 字节。

a. 为了使得 TCP 序号不至于用完，L 的最大值是多少？前面讲过 TCP 的序号字段为 4 字节。

b. 对于你在（a）中得到的 L，求出传输此文件要用多长时间？假定运输层、网络层和数据链路层首部总共为 66 字节，并加在每个报文段上，然后经 155Mbps 链路发送得到的分组。忽略流量控制和拥塞控制，使主机 A 能够一个接一个和连续不断地发送这些报文段。

P27. 主机 A 和 B 经一条 TCP 连接通信，并且主机 B 已经收到了来自 A 的最长为 126 字节的所有字节。假定主机 A 随后向主机 B 发送两个紧接着的报文段。第一个和第二个报文段分别包含了 80 字节和 40 字节的数据。在第一个报文段中，序号是 127，源端口号是 302，目的地端口号是 80。无论何时主机 B 接收到来自主机 A 的报文段，它都会发送确认。

a. 在从主机 A 发往 B 的第二个报文段中，序号、源端口号和目的端口号各是什么？

b. 如果第一个报文段在第二个报文段之前到达，在第一个到达报文段的确认中，确认号、源端口号和目的端口号各是什么？

c. 如果第二个报文段在第一个报文段之前到达，在第一个到达报文段的确认中，确认号是什么？

d. 假定由 A 发送的两个报文段按序到达 B。第一个确认丢失了而第二个确认在第一个超时间隔之后到达。画出时序图，显示这些报文段和发送的所有其他报文段和确认。（假设没有其他分组丢失。）对于图上每个报文段，标出序号和数据的字节数量；对于你增加的每个应答，标出确认号。

P28. 主机 A 和 B 直接经一条 100Mbps 链路连接。在这两台主机之间有一条 TCP 连接。主机 A 经这条连接向主机 B 发送一个大文件。主机 A 能够向它的 TCP 套接字以高达 120Mbps 的速率发送应用数据，而主机 B 能够以最大 50Mbps 的速率从它的 TCP 接收缓存中读出数据。描述 TCP 流量控制的影响。

P29. 在 3.5.6 节中讨论了 SYN cookie。

a. 服务器在 SYNACK 中使用一个特殊的初始序号，这为什么是必要的？

b. 假定某攻击者得知了一台目标主机使用了 SYN cookie。该攻击者能够通过直接向目标发送一个 ACK 分组创建半开或全开连接吗？为什么？

c. 假设某攻击者收集了由服务器发送的大量初始序号。该攻击者通过发送具有初始序号的 ACK，能够引起服务器产生许多全开连接吗？为什么？

P30. 考虑在 3.6.1 节中显示在第二种情况下的网络。假设发送主机 A 和 B 具有某些固定的超时值。

a. 证明增加路由器有限缓存的长度可能减小吞吐量（λout）。

b. 现在假设两台主机基于路由器的缓存时延，动态地调整它们的超时值（像 TCP 所做的那样）。增加缓存长度将有助于增加吞吐量吗？为什么？

P31. 假设测量的 5 个 SampleRTT 值（参见 3.5.3 节）是 106ms、120ms、140ms、90ms 和 115ms。在获得了每个 SampleRTT 值后计算 EstimatedRTT，使用 $\alpha=0.125$ 并且假设在刚获得前 5 个样本之后 EstimatedRTT 的值为 100ms。在获得每个样本之后，也计算 DevRTT，假设 $\beta=0.25$，并且假设在刚获得前 5 个样本之后 DevRTT 的值为 5ms。最后，在获得这些样本之后计算 TCP TimeoutInterval。

P32. 考虑 TCP 估计 RTT 的过程。假设 $\alpha=0.1$，令 SampleRT_{T1} 设置为最新样本 RTT，令 SampleRT_{T2} 设置为下一个最新样本 RTT，等等。

a. 对于一个给定的 TCP 连接，假定 4 个确认报文相继到达，带有 4 个对应的 RTT 值：SampleRT_{T4}、SampleRT_{T3}、SampleRT_{T2} 和 SampleRT_{T1}。根据这 4 个样本 RTT 表示 EstimatedRTT。

b. 将你得到的公式一般化到 n 个 RTT 样本的情况。

c. 对于在（b）中得到的公式，令 n 趋于无穷。试说明为什么这个平均过程被称为指数移动平均。

P33. 在 3.5.3 节中，我们讨论了 TCP 的往返时间的估计。TCP 避免测量重传报文段的 SampleRTT，对此你有何看法？

P34. 3.5.4 节中的变量 SendBase 和 3.5.5 节中的变量 LastByteRcvd 之间有什么关系？

P35. 3.5.5 节中的变量 LastByteRcvd 和 3.5.4 节中的变量 y 之间有什么关系？

P36. 在 3.5.4 节中，我们看到 TCP 直到收到 3 个冗余 ACK 才执行快速重传。你对 TCP 设计者没有选择在收到对报文段的第一个冗余 ACK 后就快速重传有何看法？

P37. 比较 GBN、SR 和 TCP（无延时的 ACK）。假设对所有 3 个协议的超时值足够长，使得 5 个连续的数据报文段及其对应的 ACK 能够分别由接收主机（主机 B）和发送主机（主机 A）收到（如果在信道中无丢失）。假设主机 A 向主机 B 发送 5 个数据报文段，并且第二个报文段（从 A 发送）丢失。最后，所有 5 个数据报文段已经被主机 B 正确接收。

a. 主机 A 总共发送了多少报文段和主机 B 总共发送了多少 ACK？它们的序号是什么？对所有 3 个协议回答这个问题。

b. 如果对所有 3 个协议超时值比 5RTT 长得多，则哪个协议在最短的时间间隔中成功地交付所有 5 个数据报文段？

P38. 在图 3-52 中的 TCP 描述中，阈值 ssthresh 的值在几个地方被设置为 ssthresh = cwnd/2，并且当出现一个丢包事件时，ssthresh 的值被设置为窗口长度的一半。当出现丢包事件时，发送方发送的速率，每个 RTT 必须大约等于 cwnd 报文段吗？解释你的答案。如果你的回答是没有，你能建议一种不同的方式，进行 ssthresh 设置吗？

P39. 考虑图 3-46b。如果 λ'_{in} 增加超过了 $R/2$，λ_{out} 能够增加超过 $R/3$ 吗？试解释之。现在考虑图 3-46c。假定一个分组从路由器到接收方平均转发两次的话，如果 λ'_{in} 增加超过 $R/2$，λ_{out} 能够增加超过 $R/4$ 吗？试解释之。

P40. 考虑图 3-58。假设 TCP Reno 是一个经历如上所示行为的协议，回答下列问题。在各种情况中，简要地论证你的回答。

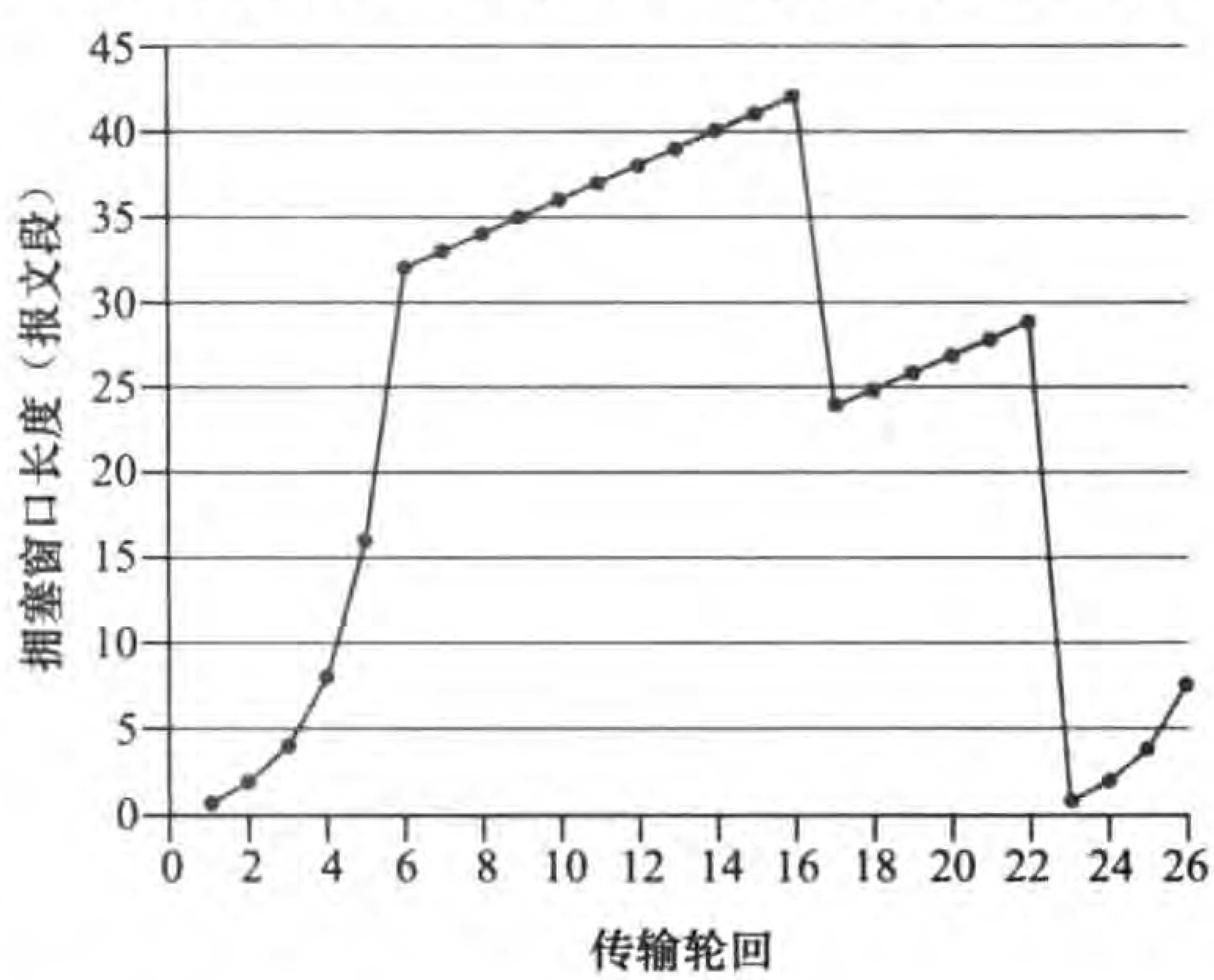


图 3-58 TCP 窗口长度作为时间的函数

a. 指出 TCP 慢启动运行时的时间间隔。

b. 指出 TCP 拥塞避免运行时的时间间隔。

c. 在第 16 个传输轮回之后，报文段的丢失是根据 3 个冗余 ACK 还是根据超时检测出来的？

d. 在第 22 个传输轮回之后，报文段的丢失是根据 3 个冗余 ACK 还是根据超时检测出来的？

e. 在第 1 个传输轮回里，ssthresh 的初始值设置为多少？

f. 在第 18 个传输轮回里，ssthresh 的值设置为多少？

g. 在第 24 个传输轮回里，ssthresh 的值设置为多少？

h. 在哪个传输轮回内发送第 70 个报文段？

i. 假定在第 26 个传输轮回后，通过收到 3 个冗余 ACK 检测出有分组丢失，拥塞的窗口长度和 ssthresh 的值应当是多少？

j. 假定使用 TCP Tahoe（而不是 TCP Reno），并假定在第 16 个传输轮回收到 3 个冗余 ACK。在第 19 个传输轮回，ssthresh 和拥塞窗口长度是什么？

k. 再次假设使用 TCP Tahoe，在第 22 个传输轮回有一个超时事件。从第 17 个传输轮回到第 22 个传输轮回（包括这两个传输轮回），一共发送了多少分组？

P41. 参考图 3-55，该图描述了 TCP 的 AIMD 算法的收敛特性。假设 TCP 不采用乘性减，而是采用按某一常量减小窗口。所得的 AIAD 算法将收敛于一种平等共享算法吗？使用类似于图 3-55 中的图来证实你的结论。

P42. 在 3.5.4 节中，我们讨论了在发生超时事件后将超时间隔加倍。为什么除了这种加倍超时间隔机制外，TCP 还需要基于窗口的拥塞控制机制（如在 3.7 节中学习的那种机制）呢？

P43. 主机 A 通过一条 TCP 连接向主机 B 发送一个很大的文件。在这条连接上，不会出现任何分组丢失和定时器超时。主机 A 与因特网连接链路的传输速率表示为 R bps。假设主机 A 上的进程能够以 S bps 的速率向 TCP 套接字发送数据，其中 $S = 10 \times R$。进一步假设 TCP 的接收缓存足够大，能够容纳整个文件，并且发送缓存只能容纳这个文件的百分之一。如何防止主机 A 上的进程连续地向 TCP 套接字以速率 S bps 传送数据呢？还是用 TCP 流量控制呢？还是用 TCP 拥塞控制？或者用其他措施？阐述其理由。

P44. 考虑从一台主机经一条没有丢包的 TCP 连接向另一台主机发送一个大文件。

a. 假定 TCP 使用不具有慢启动的 AIMD 进行拥塞控制。假设每当收到一批 ACK 时，cwnd 增加 1 个 MSS，并且假设往返时间大约恒定，cwnd 从 6MSS 增加到 12MSS 要花费多长时间（假设没有丢包事件）？

b. 对于该连接，到时间 =6RTT，其平均吞吐量是多少（根据 MSS 和 RTT）？

P45. 回想 TCP 吞吐量的宏观描述。在连接速率从 $W/(2 \times \text{RTT})$ 变化到 W/RTT 的周期内，只丢失了一个分组（在该周期的结束）。

a. 证明其丢包率（分组丢失的比率）等于：

$$L = \text{丢包率} = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

b. 如果一条连接的丢包率为 L，使用上面的结果，则它的平均速率近似由下式给出：

$$\text{平均速率} \approx \frac{1.22 * \text{MSS}}{\text{RTT}\sqrt{L}}$$

P46. 考虑仅有一条单一的 TCP（Reno）连接使用一条 10Mbps 链路，且该链路没有缓存任何数据。假设这条链路是发送主机和接收主机之间的唯一拥塞链路。假定某 TCP 发送方向接收方有一个大文件要发送，而接收方的接收缓存比拥塞窗口要大得多。我们也做下列假设：每个 TCP 报文段长度为 1500 字节；该连接的双向传播时延是 150ms；并且该 TCP 连接总是处于拥塞避免阶段，即忽略了慢启动。

a. 这条 TCP 连接能够取得的最大窗口长度（以报文段计）是多少？

b. 这条 TCP 连接的平均窗口长度（以报文段计）和平均吞吐量（以 bps 计）是多少？

c. 这条 TCP 连接在从丢包恢复后，再次到达其最大窗口要经历多长时间？

P47. 考虑在前面习题中所描述的场景。假设 10Mbps 链路能够缓存有限个报文段。试论证为了使该链路总是忙于发送数据，我们将要选择缓存长度，使得其至少为发送方和接收方之间链路速率 C 与双向传播时延之积。

P48. 重复习题 46，但用一条 10Gbps 链路代替 10Mbps 链路。注意到在对 c 部分的答案中，应当认识到在从丢包恢复后，拥塞窗口长度到达最大窗口长度将需要很长时间。给出解决该问题的基本思路。

P49. 令 T（用 RTT 度量）表示一条 TCP 连接将拥塞窗口从 $W/2$ 增加到 W 所需的时间间隔，其中 W 是最大的拥塞窗口长度。论证 T 是 TCP 平均吞吐量的函数。

P50. 考虑一种简化的 TCP 的 AIMD 算法，其中拥塞窗口长度用报文段的数量来度量，而不是用字节度量。在加性增中，每个 RTT 拥塞窗口长度增加一个报文段。在乘性减中，拥塞窗口长度减小一半（如果结果不是一个整数，向下取整到最近的整数）。假设两条 TCP 连接 C1 和 C2，它们共享一条速

率为每秒30个报文段的单一拥塞链路。假设C1和C2均处于拥塞避免阶段。连接C1的RTT是50ms，连接C2的RTT是100ms。假设当链路中的数据速率超过了链路的速率时，所有TCP连接经受数据报文段丢失。

a. 如果在时刻 t_0，C1和C2具有10个报文段的拥塞窗口，在1000ms后它们的拥塞窗口为多长？

b. 经长时间运行，这两条连接将取得共享该拥塞链路的相同的带宽吗？

P51. 考虑在前面习题中描述的网络。现在假设两条TCP连接C1和C2，它们具有相同的100ms RTT。假设在时刻 t_0，C1的拥塞窗口长度为15个报文段，而C2的拥塞窗口长度是10个报文段。

a. 在2200ms后，它们的拥塞窗口长度为多长？

b. 经长时间运行，这两条连接将取得共享该拥塞链路的相同的带宽吗？

c. 如果这两条连接在相同时间达到它们的最大窗口长度，并在相同时间达到它们的最小窗口长度，我们说这两条连接是同步的。经长时间运行，这两条连接将最终变得同步吗？如果是，它们的最大窗口长度是多少？

d. 这种同步将有助于改善共享链路的利用率吗？为什么？给出打破这种同步的某种思路。

P52. 考虑修改TCP的拥塞控制算法。不使用加性增，使用乘性增。无论何时某TCP收到一个合法的ACK，就将其窗口长度增加一个小正数 a $(0<a<1)$。求出丢包率 L 和最大拥塞窗口 W 之间的函数关系。论证：对于这种修正的TCP，无论TCP的平均吞吐量如何，一条TCP连接将其拥塞窗口长度从 $W/2$ 增加到 W，总是需要相同的时间。

P53. 在3.7节对TCP未来的讨论中，我们注意到为了取得10Gbps的吞吐量，TCP仅能容忍 2×10^{-10} 的报文段丢失率（或等价为每5 000 000 000个报文段有一个丢包事件）。给出针对3.7节中给定的RTT和MSS值的对 2×10^{-10} 值的推导。如果TCP需要支持一条100Gbps的连接，所能容忍的丢包率是多少？

P54. 在3.7节中对TCP拥塞控制的讨论中，我们隐含地假定TCP发送方总是有数据要发送。现在考虑下列情况，某TCP发送方发送大量数据，然后在 t_1 时刻变得空闲（因为它没有更多的数据要发送）。TCP在相对长的时间内保持空闲，然后在 t_2 时刻要发送更多的数据。当TCP在 t_2 开始发送数据时，让它使用在 t_1 时刻的cwnd和ssthresh值，将有什么样的优点和缺点？你建议使用什么样的方法？为什么？

P55. 在这个习题中我们研究是否UDP或TCP提供了某种程度的端点鉴别。

a. 考虑一台服务器接收到在一个UDP分组中的请求并对该请求进行响应（例如，如由DNS服务器所做的那样）。如果一个具有IP地址X的客户用地址Y进行哄骗的话，服务器将向何处发送它的响应？

b. 假定一台服务器接收到具有IP源地址Y的一个SYN，在用SYNACK响应之后，接收一个具有IP源地址Y和正确确认号的ACK。假设该服务器选择了一个随机初始序号并且没有“中间人”，该服务器能够确定该客户的确位于Y吗？（并且不在某个其他哄骗为Y的地址X。）

P56. 在这个习题中，我们考虑由TCP慢启动阶段引入的时延。考虑一个客户和一个Web服务器直接连接到速率 R 的一条链路。假定该客户要取回一个对象，其长度正好等于 $15S$，其中 S 是最大段长度（MSS）。客户和服务器之间的往返时间表示为RTT（假设为常数）。忽略协议首部，确定在下列情况下取回该对象的时间（包括TCP连接创建）：

a. $4S/R > S/R + \mathrm{RTT} > 2S/R$

b. $S/R + \mathrm{RTT} > 4S/R$

c. $S/R > \mathrm{RTT}$

编程作业

实现一个可靠运输协议

在这个编程作业实验中，你将要编写发送和接收运输层的代码，以实现一个简单的可靠数据运输协

议。这个实验有两个版本，即比特交替协议版本和 GBN 版本。这个实验应当是有趣的，因为你的实现将与实际情况下所要求的差异很小。

因为可能没有你能够修改其操作系统的独立机器，你的代码将不得不在模拟的硬件/软件环境中执行。然而，为你提供例程的编程接口（即从上层和下层调用你的实体的代码），非常类似于在实际 UNIX 环境中做那些事情的接口。（实际上，在本编程作业中描述的软件接口比起许多教科书中描述的无限循环的发送方和接收方要真实得多。）停止和启动定时器也是模拟的，定时器中断将激活你的定时器处理例程。

这个完整的实验作业以及你所需要的代码，可从本书的 Web 网站上获得：http://www.pearsonhighered.com/cs-resources。

Wireshark 实验：探究 TCP

在这个实验中，你将使用 Web 浏览器访问来自某 Web 服务器的一个文件。如同在前面的 Wireshark 实验中一样，你将使用 Wireshark 来俘获到达你计算机的分组。与前面实验不同的是，你也能够从该 Web 服务器下载一个 Wireshark 可读的分组踪迹，记载你从服务器下载文件的过程。在这个服务器踪迹文件里，你将发现自己访问该 Web 服务器所产生的分组。你将分析客户端和服务器端踪迹文件，以探究 TCP 的方方面面。特别是你将评估在你的计算机与该 Web 服务器之间 TCP 连接的性能。你将跟踪 TCP 窗口行为、推断分组丢失、重传、流控和拥塞控制行为并估计往返时间。

与所有的 Wireshark 实验一样，该实验的全面描述能够在本书 Web 站点 http://www.pearsonhighered.com/cs-resources 上找到。

Wireshark 实验：探究 UDP

在这个简短实验中，你将进行分组俘获并分析那些使用 UDP 的你喜爱的应用程序（例如，DNS 或如 Skype 这样的多媒体应用）。如我们在 3.3 节中所学的那样，UDP 是一种简单的、不提供不必要服务的运输协议。在这个实验中，你将研究在 UDP 报文段中的各首部字段以及检验和计算。

与所有的 Wireshark 实验一样，该实验的全面描述能够在本书 Web 站点 http://www.pearsonhighered.com/cs-resources 上找到。

人物专访

Van Jacobson 就职于谷歌公司，以前是 PARC 的高级研究员。在此之前，他是分组设计（Packet Design）组织的共同创建者和首席科学家。再往前，他是思科公司的首席科学家。在加入思科之前，他是劳伦兹伯克利国家实验室的网络研究组的负责人，并在加州大学伯克利分校和斯坦福大学任教。Van 于 2001 年因其在通信网络领域的贡献而获得 ACM SIGCOMM 终身成就奖，于 2002 年因其“对网络拥塞的理解和成功研制用于因特网的拥塞控制机制”而获得 IEEE Kobayashi 奖。他于 2004 年当选为美国国家工程院院士。



Van Jacobson

- 请描述您职业生涯中做过的一两个最令人激动的项目。最大的挑战是什么？

学校教会我们许多寻找答案的方式。在每个我致力于的感兴趣的问题中，艰巨的任务是找到正确的问题。当 Mike Karels 和我开始关注 TCP 拥塞时，我们花费数月凝视协议和分组踪迹，询问“为什么它会失效?”。有一天在 Mike 的办公室，我们中的一个说：“我无法弄明白它失效的原因是因为我不理解它究竟如何开始运转的。”这导致提出了一个正确问题，它迫使我们弄明白使 TCP 运转的“ack 计时”。从那以后，其他东西就容易了。

- 从更为一般的意义上讲，您认为网络和因特网未来将往何处发展?

对于大多数人来说，Web 是因特网。对此，网络奇才们将会善意地窃笑，因为我们知道 Web 是一个

运行在因特网上的应用程序，但要是以上说法正确又该如何呢？因特网使得主机对之间能够进行交谈。Web 用于分布信息的生产和消耗。“信息传播”是一种非常一般意义上的通信，而“成对交谈”只是其中一个极小的子集。我们需要向更大的范围进发。今天的网络以点到点连线的方式处理广播媒体（无线电、PON 等）。那是极为低效的。经过指头敲击或智能手机，遍及全世界的每秒兆兆（10^{12}）比特的数据正在交换，但我们不知道如何将其作为“网络”处理。ISP 正在忙于建立缓存和 CDN，以可扩展地分发视频和音频。缓存是该解决方案的必要部分，但今天的网络缺乏这个部分。从信息论、排队论或流量理论直到因特网协议规范，都告诉我们如何建造和部署它。我认为并希望在未来几年中，网络将演化为包含多得多的通信愿景，支撑 Web 的运行。

- 是谁激发了您的职业灵感？

当我还在研究生院时，Richard Feynman 访问了学校并做了学术报告。他讲到了一些量子理论知识，使我整学期都在努力理解该理论，他的解释非常简单和明白易懂，使得那些对我而言难以理解的东西变得显而易见和不可避免。领会和表达复杂世界背后的简单性的能力是给我的罕见和绝妙的礼物。

- 您对进入计算机科学和网络领域的学生有什么忠告吗？

网络是奇妙的领域，计算机和网络对社会的影响，也许比自有文字记载以来的任何发明都大。网络本质上是有关连接的东西，研究它有助于你进行智能连接：蚁群搜索和蜜蜂舞蹈显示了协议设计好于 RFC，流量拥挤或人们离开挤满人的体育馆是拥塞的要素，在感恩节暴风雪后寻找航班返回学校的学生们是动态路由选择的核心。如果你对许多东西感兴趣，并且要对此干点事，很难想象还有什么比网络更好的领域了。

网络层：数据平面

在前一章中我们学习了运输层依赖于网络层的主机到主机的通信服务，提供各种形式的进程到进程的通信。我们也学习了运输层工作时不具备任何有关网络层实际实现这种服务的知识。因此也许你现在想知道，这种主机到主机通信服务的真实情况是什么？是什么使得它工作起来的呢？

在本章和下一章中，我们将学习网络层实际是怎样实现主机到主机的通信服务的。我们将看到与运输层和应用层不同的是，在网络中的每一台主机和路由器中都有一个网络层部分。正因如此，网络层协议是协议栈中最具挑战性（因而也是最有趣）的部分。

网络层在协议栈中毋庸置疑是最复杂的层次，因此我们将在这里用大量篇幅来讨论。的确因为涉及的内容太多，我们要用两章的篇幅来讨论网络层。我们将看到网络层能够被分解为两个相互作用的部分，即**数据平面**和**控制平面**。在第4章，我们将首先学习网络层的数据平面功能，即网络层中每台路由器的功能，该数据平面功能决定到达路由器输入链路之一的数据报（即网络层的分组）如何转发到该路由器的输出链路之一。我们将涉及传统的IP转发（其中转发基于数据报的目的地址）和通用的转发（其中可以使用数据报首部中的几个不同域的值执行转发和其他功能）。我们将详细地学习IPv4和IPv6协议及其寻址。在第5章，我们将涉及网络层的控制平面功能，即网络范围的逻辑，该控制平面功能控制数据报沿着从源主机到目的主机的端到端路径中路由器之间的路由方式。我们将学习路由选择算法，以及广泛用于今天因特网中的诸如OSPF和BGP等路由选择协议。传统上，这些控制平面路由选择协议和数据平面转发功能已被实现成一个整体，位于一台路由器中。软件定义网络（Software-Defined Networking，SDN）通过将这些控制平面功能作为一种单独服务，明确地分离数据平面和控制平面，控制平面功能通常置于一台远程“控制器”中。我们将在第5章涉及SDN控制器。

网络层中数据平面和控制平面之间的功能区别很重要，当你学习网络层时，心中要记住这个区别。它将有助于你构思网络层，并且反映计算机网络中网络层角色的现代观点。

4.1 网络层概述

图4-1显示了一个简单网络，其中有H1和H2两台主机，在H1与H2之间的路径上有几台路由器。假设H1正在向H2发送信息，考虑这些主机与中间路由器的网络层所起的作用。H1中的网络层取得来自于H1运输层的报文段，将每个报文段封装成一个数据报，然后向相邻路由器R1发送该数据报。在接收方主机H2，网络层接收来自相邻路由器R2的数据报，提取出运输层报文段，并将其向上交付给H2的运输层。每台路由器的数据平面的主要作用是从其输入链路向其输出链路转发数据报；控制平面的主要作用是协调这些本地的每路由器转发动作，使得数据报沿着源和目的地主机之间的路由器路径最终进行端到端传送。注意到图4-1中所示路由器具有截断的协议栈，即没有网络层以上的部分，

因为路由器不运行我们已在第2、3章学习过的应用层和运输层协议。

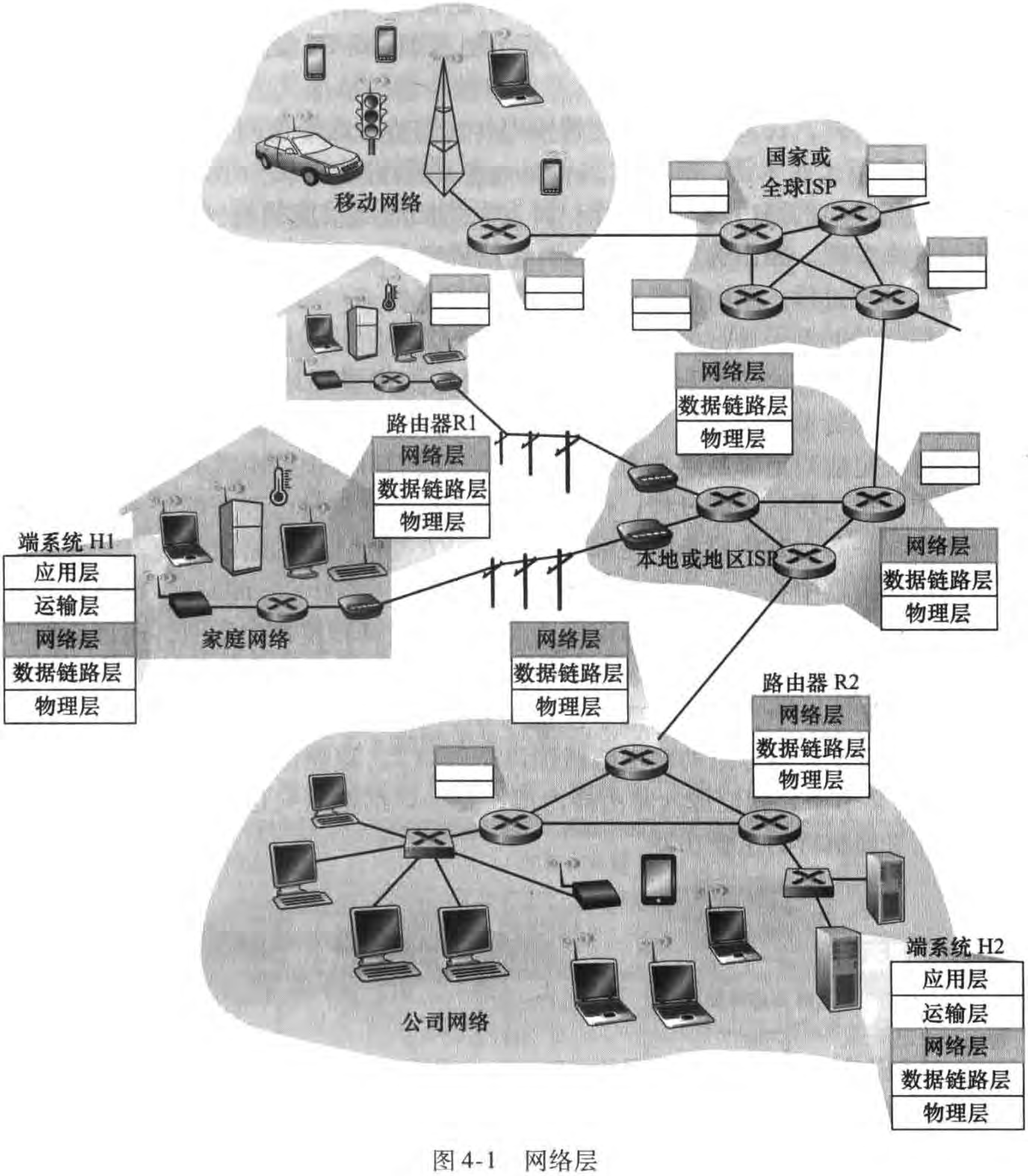


图 4-1　网络层

4.1.1　转发和路由选择：数据平面和控制平面

网络层的作用从表面上看极为简单，即将分组从一台发送主机移动到一台接收主机。为此，需要使用两种重要的网络层功能：

- 转发。当一个分组到达某路由器的一条输入链路时，该路由器必须将该分组移动到适当的输出链路。例如，在图4-1中来自主机H1到路由器R1的一个分组，必须向到达H2的路径上的下一台路由器转发。如我们将看到的那样，转发是在数据平面中实现的唯一功能（尽管是最为常见和重要的功能）。在最为常见的场合（我们将在4.4节中讨论），分组也可能被现有的路由器阻挡（例如，该分组来源于一个已知的恶意主机，或者该分组发向一个被禁止的目的主机），或者可能是冗余的并经过多条出链路发送。

- **路由选择**。当分组从发送方流向接收方时，网络层必须决定这些分组所采用的路由或路径。计算这些路径的算法被称为**路由选择算法**（routing algorithm）。例如，在图 4-1 中一个路由选择算法将决定分组从 H1 到 H2 流动所遵循的路径。路由选择在网络层的控制平面中实现。

在讨论网络层时，许多作者经常交替使用*转发*和*路由选择*这两个术语。我们在本书中将更为精确地使用这些术语。**转发**（forwarding）是指将分组从一个输入链路接口转移到适当的输出链路接口的路由器本地动作。转发发生的时间尺度很短（通常为几纳秒），因此通常用硬件来实现。**路由选择**（routing）是指确定分组从源到目的地所采取的端到端路径的网络范围处理过程。路由选择发生的时间尺度长得多（通常为几秒），因此通常用软件来实现。用驾驶的例子进行类比，考虑在 1.3.1 节中旅行者所历经的从宾夕法尼亚州到佛罗里达州的行程。在这个行程中，那位驾驶员在到佛罗里达州的途中经过了许多立交桥。我们能够认为转发就像通过单个立交桥的过程：一辆汽车从其道路上进入立交桥的一个入口，并且决定应当走哪条路来离开该立交桥。我们可以把路由选择看作是规划从宾夕法尼亚州到佛罗里达州行程的过程：在着手行程之前，驾驶员已经查阅了地图并在许多可能的路径中选择一条，其中每条路径都由一系列经立交桥连接的路段组成。

每台网络路由器中有一个关键元素是它的**转发表**（forwarding table）。路由器检查到达分组首部的一个或多个字段值，进而使用这些首部值在其转发表中索引，通过这种方法来转发分组。这些值对应存储在转发表项中的值，指出了该分组将被转发的路由器的输出链路接口。例如在图 4-2 中，一个首部字段值为 0111 的分组到达路由器。该路由器在它的转发表中索引，并确定该分组的输出链路接口是接口 2。该路由器则在内部将该分组转发到接口 2。在 4.2 节中，我们深入路由器内部，更为详细地研究这种转发功能。转发是由网络层的数据平面执行的主要功能。

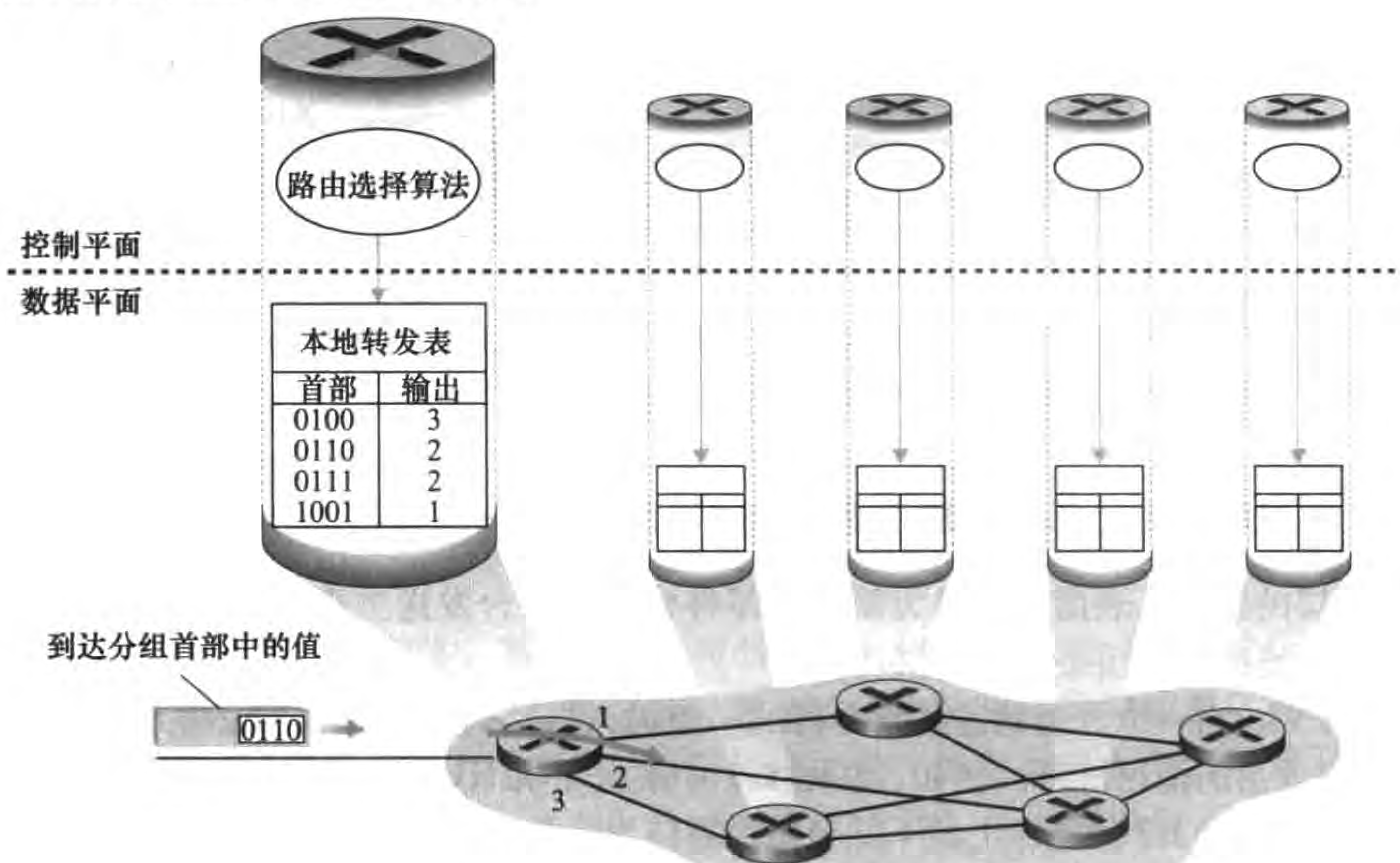


图 4-2　路由选择算法决定转发表中的值

1. 控制平面：传统的方法

你也许现在想知道路由器中的转发表一开始是如何配置的。这是一个关键问题，它揭

示了路由选择和转发间的重要相互作用关系。如图4-2所示，路由选择算法决定了插入该路由器转发表的内容。在这个例子中，路由选择算法运行在每台路由器中，并且在每台路由器中都包含转发和路由选择两种功能。如我们将在5.3节和5.4节中所见，在一台路由器中的路由选择算法与在其他路由器中的路由选择算法通信，以计算出它的转发表的值。这种通信是如何执行的呢？通过根据路由选择协议交换包含路由选择信息的路由选择报文！我们将在5.2～5.4节讨论路由选择算法和协议。

通过考虑网络中的假想情况（不真实的，但技术上是可行的），也就是说路由器中物理上存在的所有转发表的内容是由人类网络操作员直接配置的，进一步说明转发和路由选择功能的区别和不同目的。在这种情况下，不需要任何路由选择协议！当然，这些人类操作员将需要彼此交互，以确保该转发表的配置能使分组到达它们想要到达的目的地。出现下列现象也很可能：人工配置更容易出错，并且对于网络拓扑变化的响应比起路由选择协议来更慢。我们要为所有网络具有转发和路由选择功能而感到幸运！

2. 控制平面：SDN方法

图4-2中显示的实现路由选择功能的方法，是路由选择厂商在其产品中采用的传统方法，至少最近还是如此。使用该方法，每台路由器都有一个与其他路由器的路由选择组件通信的路由选择组件。然而，对人类能够手动配置转发表的观察启发我们，对于控制平面功能来说，也许存在其他方式来确定数据平面转发表的内容。

图4-3显示了从路由器物理上分离的另一种方法，远程控制器计算和分发转发表以供每台路由器所使用。注意到图4-2和图4-3的数据平面组件是相同的。而在图4-3中，控制平面路由选择功能与物理的路由器是分离的，即路由选择设备仅执行转发，而远程控制

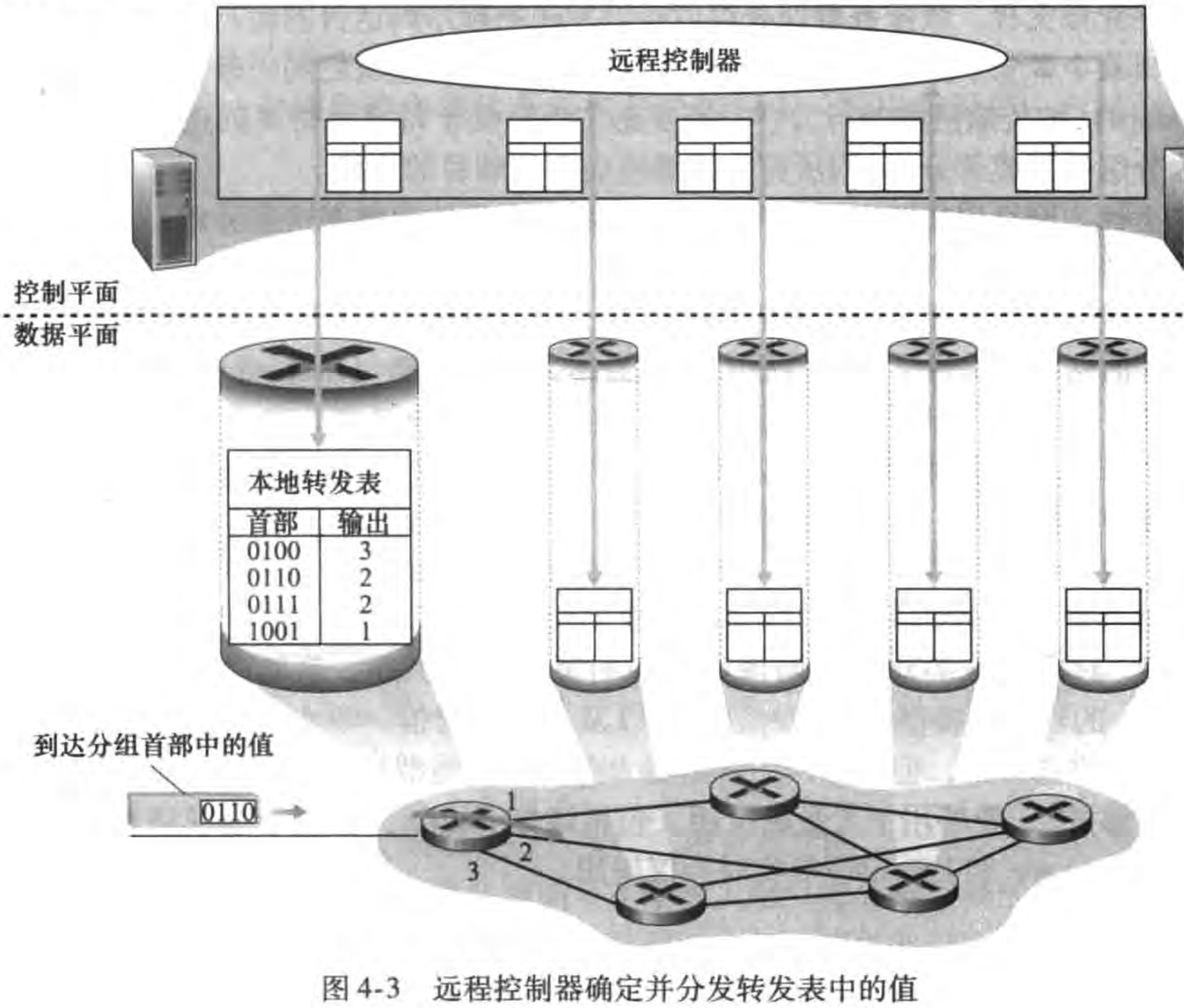


图4-3　远程控制器确定并分发转发表中的值

器计算并分发转发表。远程控制器可能实现在具有高可靠性和冗余的远程数据中心中，并可能由ISP或某些第三方管理。路由器和远程控制器是如何通信的呢？通过交换包含转发表和其他路由选择信息的报文。显示在图4-3中的控制平面方法是**软件定义网络**（Software-Defined Networking，SDN）的本质，因为计算转发表并与路由器交互的控制器是用软件实现的，故网络是“软件定义”的。这些软件实现也越来越开放，换言之类似于Linux操作系统代码，这些代码可为公众所用，允许ISP（以及网络研究者和学生）去创新并对控制网络层功能的软件提出更改建议。我们将在5.5节中讨论SDN控制平面。

4.1.2 网络服务模型

在钻研网络层的数据平面之前，我们将以开阔的视野来专注于我们引入的新东西并考虑网络层可能提供的不同类型的服务。当位于发送主机的运输层向网络传输分组时（即在发送主机中将分组向下交给网络层），运输层能够指望网络层将该分组交付给目的地吗？当发送多个分组时，它们会按发送顺序按序交付给接收主机的运输层吗？发送两个连续分组的时间间隔与接收到这两个分组的时间间隔相同吗？网络层会提供关于网络中拥塞的反馈信息吗？在发送主机与接收主机中连接运输层通道的抽象视图（特性）是什么？对这些问题和其他问题的答案由网络层提供的服务模型所决定。**网络服务模型**（network service model）定义了分组在发送与接收端系统之间的端到端运输特性。

我们现在考虑网络层能提供的某些可能的服务。这些服务可能包括：

- *确保交付*。该服务确保分组将最终到达目的地。
- *具有时延上界的确保交付*。该服务不仅确保分组的交付，而且在特定的主机到主机时延上界内（例如在100ms内）交付。
- *有序分组交付*。该服务确保分组以它们发送的顺序到达目的地。
- *确保最小带宽*。这种网络层服务模仿在发送和接收主机之间一条特定比特率（例如1Mbps）的传输链路的行为。只要发送主机以低于特定比特率的速率传输比特（作为分组的组成部分），则所有分组最终会交付到目的主机。
- *安全性*。网络层能够在源加密所有数据报并在目的地解密这些分组，从而对所有运输层报文段提供机密性。

这只是网络层能够提供的服务的部分列表，有无数种可能的服务变种。

因特网的网络层提供了单一的服务，称为**尽力而为服务**（best-effort service）。使用尽力而为服务，传送的分组既不能保证以它们发送的顺序被接收，也不能保证它们最终交付；既不能保证端到端时延，也不能保证有最小的带宽。*尽力而为服务看起来是根本无服务*的一种委婉说法，即一个没有向目的地交付分组的网络也符合尽力而为交付服务的定义！其他的网络体系结构已定义和实现了超过因特网尽力而为服务的服务模型。例如，ATM网络体系结构［MFA Forum 2016；Black 1995］提供了确保按序时延、有界时延和确保最小带宽。还有提议的对因特网体系结构的服务模型扩展，例如，集成服务体系结构［RFC 1633］的目标是提供端到端时延保证以及无拥塞通信。令人感兴趣的是，尽管有这些研发良好的供选方案，但因特网的基本尽力而为服务模型与适当带宽供给相结合已被证明超过“足够好”，能够用于大量的应用，包括诸如Netflix、IP语音和视频等流式视频服务，以及诸如Skype和Facetime等实时会议应用。

第4章概述

在提供了网络层的概述后，我们将在本章后续几节中讨论网络层的数据平面组件。在

4.2 节中，我们将深入探讨路由器的内部硬件操作，包括输入和输出分组处理、路由器的内部交换机制以及分组排队和调度。在 4.3 节中，我们将学习传统的 IP 转发，其中分组基于它们的目的 IP 地址转发到输出端口。我们将学习到 IP 寻址、令人称道的 IPv4 和 IPv6 协议等。在 4.4 节中，我们将涉及更为一般的转发，此时分组可以基于大量首部值（即不仅基于目的 IP 地址）转发到输出端口。分组可能在路由器中受阻或冗余，或者可能让某些首部字段重写，即所有都在软件控制之下完成。这种分组转发的更为一般的形式是现代网络数据平面的关键组件，包括软件定义网络（SDN）中的数据平面。

我们在这里顺便提到，许多计算机网络研究者和从业人员经常互换地使用转发和交换这两个术语。我们在这本教科书中也将互换使用这些术语。在我们开始讨论术语的主题时，还需要指出经常互换使用的两个其他术语，但我们将更为小心地使用它们。我们将约定术语分组交换机是指一台通用分组交换设备，它根据分组首部字段中的值，从输入链路接口到输出链路接口转移分组。某些分组交换机称为**链路层交换机**（link-layer switch）（在第 6 章仔细学习），基于链路层帧中的字段值做出转发决定，这些交换机因此被称为链路层（第 2 层）设备。其他分组交换机称为**路由器**（router），基于网络层数据报中的首部字段值做出转发决定。路由器因此是网络层（第 3 层）设备。（为了全面理解这种重要区别，你可能要回顾 1.5.2 节，在那里我们讨论了网络层数据报和链路层帧及其关系。）因为在本章中我们关注的是网络层，所以我们将主要使用术语路由器来代替交换机。

4.2 路由器工作原理

既然我们已经概述了网络层中的数据平面和控制平面、转发与路由选择之间的重要区别以及网络层的服务与功能，我们将注意力转向网络层的**转发功能**，即实际将分组从一台路由器的入链路传送到适当的出链路。

图 4-4 显示了一个通用路由器体系结构的总体视图，其中标识了一台路由器的 4 个组件。

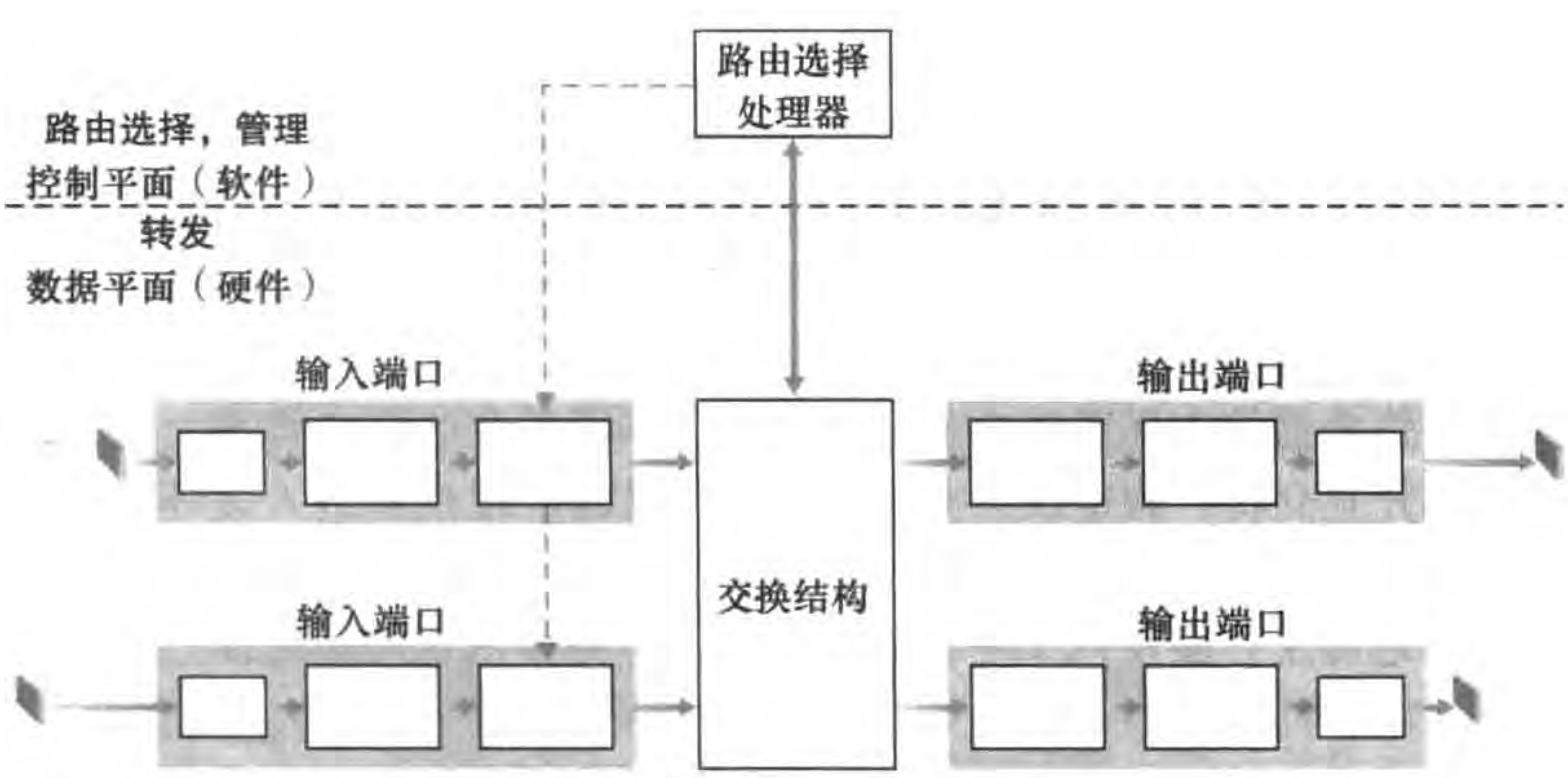


图 4-4 路由器体系结构

- 输入端口。**输入端口**（input port）执行几项重要功能。它在路由器中执行终结入物理链路的物理层功能，这显示在图 4-4 中输入端口部分最左侧的方框与输出端口部分最右侧的方框中。它还要与位于入链路远端的数据链路层交互来执行数据链路层功能，这显示在输入与输出端口部分中间的方框中。也许更为重要的是，在输入端

口还要执行查找功能，这显示在输入端口最右侧的方框中。正是在这里，通过查询转发表决定路由器的输出端口，到达的分组通过路由器的交换结构转发到输出端口。控制分组（如携带路由选择协议信息的分组）从输入端口转发到路由选择处理器。注意这里的“端口”一词，指的是路由器的物理输入和输出接口，这完全不同于第 2、3 章中所讨论的与网络应用程序和套接字相关联的软件端口。在实践中，一台路由器所支持的端口数量范围较大，从企业路由器具有数量相对少的端口，到位于某 ISP 边缘的路由器具有数以百计 10Gbps 端口（其中入线路的数量趋于最大）。例如，边缘路由器 Juniper MX2020 具有 80Tbps 的总体路由器系统容量，支持多达 960 个 10Gbps 以太网端口 [Juniper MX 2020 2016]。

- **交换结构**。交换结构将路由器的输入端口连接到它的输出端口。这种交换结构完全包含在路由器之中，即它是一个网络路由器中的网络！
- **输出端口**。输出端口存储从交换结构接收的分组，并通过执行必要的链路层和物理层功能在输出链路上传输这些分组。当一条链路是双向的时（即承载两个方向的流量），输出端口通常与该链路的输入端口成对出现在同一线路卡上。
- **路由选择处理器**。路由选择处理器执行控制平面功能。在传统的路由器中，它执行路由选择协议（我们将在 5.3 节和 5.4 节学习），维护路由选择表与关联链路状态信息，并为该路由器计算转发表。在 SDN 路由器中，路由选择处理器（在其他活动中）负责与远程控制器通信，目的是接收由远程控制器计算的转发表项，并在该路由器的输入端口安装这些表项。路由选择处理器还执行网络管理功能，我们将在 5.7 节学习相关内容。

路由器的输入端口、输出端口和交换结构几乎总是用硬件实现，如图 4-4 所示。为了理解为何需要用硬件实现，考虑具有 10Gbps 输入链路和 64 字节的 IP 数据报，其输入端口在另一个数据报到达前仅有 51.2ns 来处理数据报。如果 N 个端口结合在一块线路卡上（因为实践中常常这样做），数据报处理流水线必须以 N 倍速率运行，这远快过软件实现的速率。转发硬件既能够使用路由器厂商自己的硬件设计来实现，也能够使用购买的商用硅片（例如由英特尔和 Broadcom 公司所出售）的硬件设计来实现。

当数据平面以纳秒时间尺度运行时，路由器的控制功能以毫秒或秒时间尺度运行，这些控制功能包括执行路由选择协议、对上线或下线的连接链路进行响应、与远程控制器通信（在 SDN 场合）和执行管理功能。因而这些**控制平面**（control plane）的功能通常用软件实现并在路由选择处理器（通常是一种传统的 CPU）上执行。

在深入探讨路由器的内部细节之前，我们转向本章开头的那个类比，其中分组转发好比汽车进入和离开立交桥。假定该立交桥是环状交叉路，在汽车进入该环状交叉路前，需要做一点处理。我们来考虑一下对于这种处理需要什么信息。

- **基于目的地转发**。假设汽车停在一个入口站上并指示它的最终目的地（并非在本地环状交叉路，而是其旅途的最终目的地）。入口站的一名服务人员查找最终目的地，决定通向最后目的地的环状交叉路的出口，并告诉驾驶员要走哪个出口。
- **通用转发**。除了目的地之外，服务人员也能够基于许多其他因素确定汽车的出口匝道。例如，所选择的出口匝道可能与该汽车的起点如发行该车牌照的州有关。来自某些州的汽车可能被引导使用某个出口匝道（经过一条慢速道路通向目的地），而来自其他州的汽车可能被引导使用一个不同的出口匝道（经过一条高速路通向目的地）。基于汽车的模型、品牌和寿命，可能做出相同的决定。或者认为不适合上路

的汽车可能被阻止并且不允许通过环状交叉路。就通用转发来说，许多因素都会对服务人员为给定汽车选择出口匝道产生影响。

一旦汽车进入环状交叉路（该环状交叉路可能挤满了从其他输入道路进入的其他汽车，朝着其他环状交叉路出口前进），并且最终离开预定的环状交叉路出口匝道，在这里可能遇到了从该出口离开环状交叉路的其他汽车。

在这个类比中，我们能够在图4-4中识别最重要的路由器组件：入口道路和入口站对应于输入端口（具有查找功能以决定本地输出端口）；环状交叉路对应于交换结构；环状交叉路出口匝道对应于输出端口。借助于这个类比，我们可以考虑瓶颈可能出现的地方。如果汽车以极快的速率到达（例如，该环状交叉路位于德国或意大利！）而车站服务人员很慢，将发生什么情况？这些服务人员必须工作得多快，以确保在入口路上没有车辆拥堵？甚至对于极快的服务人员，如果汽车在环状交叉路上开得很慢，将发生什么情况，拥堵仍会出现吗？如果大多数进入的汽车都要在相同的出口匝道离开环状交叉路，将发生什么情况，在出口匝道或别的什么地方会出现拥堵吗？如果我们要为不同的汽车分配优先权，或先行阻挡某些汽车进入环状交叉路，环状交叉路将如何运行？这些全都与路由器和交换机设计者面对的问题形成类比。

在下面的各小节中，我们将更为详细地考察路由器功能。[Iyer 2008；Chao 2001；Chuang 2005；Turner 1988；McKeown 1997a；Partridge 1998；Sopranos 2011] 提供了对一些特定路由器体系结构的讨论。为了具体和简单起见，我们在本节中初始假设转发决定仅基于分组的目的地址，而非基于通用的分组首部字段。我们将在4.4节中学习更为通用的分组转发情况。

4.2.1 输入端口处理和基于目的地转发

图4-5中显示了一个更详细的输入处理的视图。如前面讨论的那样，输入端口的线路端接功能与链路层处理实现了用于各个输入链路的物理层和链路层。在输入端口中执行的查找对于路由器运行是至关重要的。正是在这个地方，路由器使用转发表来查找输出端口，使得到达的分组能经过交换结构转发到该输出端口。转发表是由路由选择处理器计算和更新的（使用路由选择协议与其他网络路由器中的路由选择处理器进行交互），或者转发表接收来自远程SDN控制器的内容。转发表从路由选择处理器经过独立总线（例如一个PCI总线）复制到线路卡，在图4-4中该总线由从路由选择处理器到输入线路卡的虚线所指示。使用在每个输入端口的影子副本，转发决策能在每个输入端口本地做出，无须基于每个分组调用集中式路由选择处理器，因此避免了集中式处理的瓶颈。

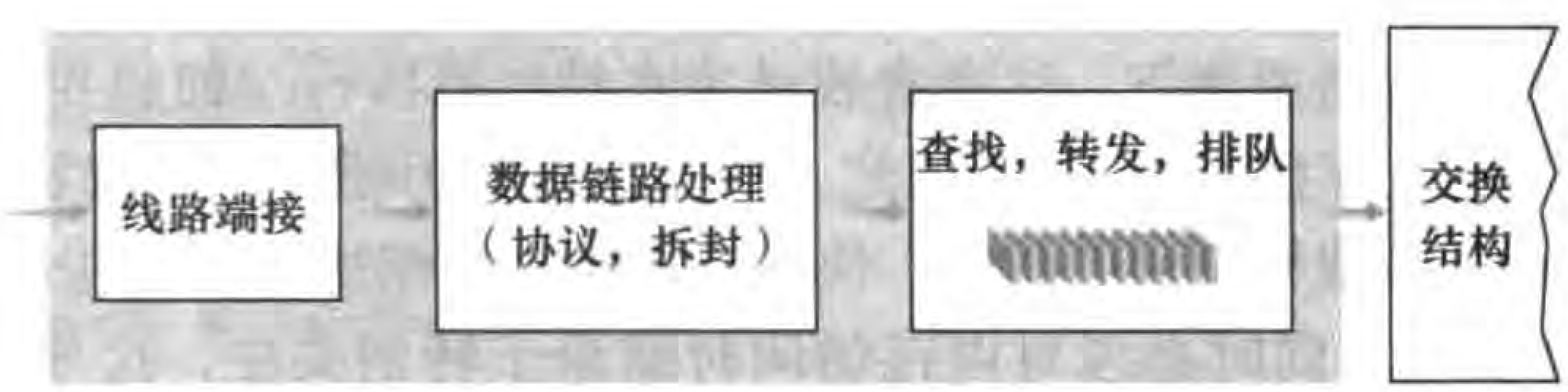


图4-5 输入端口处理

现在我们来考虑“最简单”的情况，一个入分组基于该分组的目的地址交换到输出端口。在32比特IP地址的情况下，转发表的蛮力实现将针对每个目的地址有一个表项。因

为有超过40亿个可能的地址，选择这种方法总体上是不可行的。

作为一个说明怎样处理规模问题的例子，假设我们的路由器具有4条链路，编号0到3，分组以如下方式转发到链路接口：

目的地址范围	链路接口
11001000 00010111 00010000 00000000 到 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 到 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 到 11001000 00010111 00011111 11111111	2
其他	3

显然，对于这个例子，在路由器的转发表中没有必要有40亿个表项。例如，我们能够有一个如下仅包括4个表项的转发表：

前缀匹配	链路接口
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
其他	3

使用这种风格的转发表，路由器用分组目的地址的**前缀**（prefix）与该表中的表项进行匹配；如果存在一个匹配项，则路由器向与该匹配项相关联的链路转发分组。例如，假设分组的目的地址是11001000 00010111 00010110 10100001，因为该地址的21比特前缀匹配该表的第一项，所以路由器向链路接口0转发该分组。如果一个前缀不匹配前3项中的任何一项，则路由器向链路接口3转发该分组。尽管听起来足够简单，但这里还是有重要的微妙之处。你可能已经注意到一个目的地址可能与不止一个表项相匹配。例如，地址11001000 00010111 00011000 10101010的前24比特与表中的第二项匹配，而该地址的前21比特与表中的第三项匹配。当有多个匹配时，该路由器使用**最长前缀匹配规则**（longest prefix matching rule）；即在该表中寻找最长的匹配项，并向与最长前缀匹配相关联的链路接口转发分组。当在4.3节中详细学习因特网编址时，我们将完全明白使用这种最长前缀匹配规则的理由。

假定转发表已经存在，从概念上讲表查找是简单的，硬件逻辑只是搜索转发表查找最长前缀匹配。但在吉比特速率下，这种查找必须在纳秒级执行（回想我们前面10Gbps链路和一个64字节IP数据报的例子）。因此，不仅必须要用硬件执行查找，而且需要对大型转发表使用超出简单线性搜索的技术；快速查找算法的综述能够在［Gupta 2001，Ruiz-Sanchez 2011］中找到。同时必须对内存访问时间给予特别关注，这导致用嵌入式片上DRAM和更快的SRAM（用作一种DRAM缓存）内存来设计。实践中也经常使用**三态内容可寻址存储器**（Tenary Content Address Memory，TCAM）来查找［Yu 2004］。使用TCAM，一个32比特IP地址被放入内存，TCAM在基本常数时间内返回对该地址的转发表项的内容。Cisco Catalyst 6500和7500系列路由器及交换机能够保存100多万TCAM转发表项

[Cisco TCAM 2014]。

一旦通过查找确定了某分组的输出端口，则该分组就能够发送进入交换结构。在某些设计中，如果来自其他输入端口的分组当前正在使用该交换结构，一个分组可能会在进入交换结构时被暂时阻塞。因此，一个被阻塞的分组必须要在输入端口处排队，并等待稍后被及时调度以通过交换结构。我们稍后将仔细观察分组（位于输入端口与输出端口中）的阻塞、排队与调度。尽管“查找”在输入端口处理中可认为是最为重要的动作，但必须采取许多其他动作：①必须出现物理层和链路层处理，如前面所讨论的那样；②必须检查分组的版本号、检验和以及寿命字段（这些我们将在4.3节中学习），并且重写后两个字段；③必须更新用于网络管理的计数器（如接收到的IP数据报的数目）。

在结束输入端口处理的讨论之前，注意到输入端口查找目的IP地址（“匹配”），然后发送该分组进入交换结构（“动作”）的步骤是一种更为一般的“匹配加动作”抽象的特定情况，这种抽象在许多网络设备中执行，而不仅在路由器中。在链路层交换机（在第6章讨论）中，除了发送帧进入交换结构去往输出端口外，还要查找链路层目的地址，并采取几个动作。在防火墙（在第8章讨论）中，首部匹配给定准则（例如源/目的IP地址和运输层端口号的某种组合）的入分组可能被阻止转发，而防火墙是一种过滤所选择的入分组的设备。在网络地址转换器（NAT，在4.3节讨论）中，一个运输层端口号匹配某给定值的入分组，在转发（动作）前其端口号将被重写。的确，“匹配加动作”抽象不仅作用大，而且在网络设备中无所不在，并且对于我们将在4.4节中学习的通用转发是至关重要的。

4.2.2 交换

交换结构位于一台路由器的核心部位，因为正是通过这种交换结构，分组才能实际地从一个输入端口交换（即转发）到一个输出端口中。交换可以用许多方式完成，如图4-6所示。

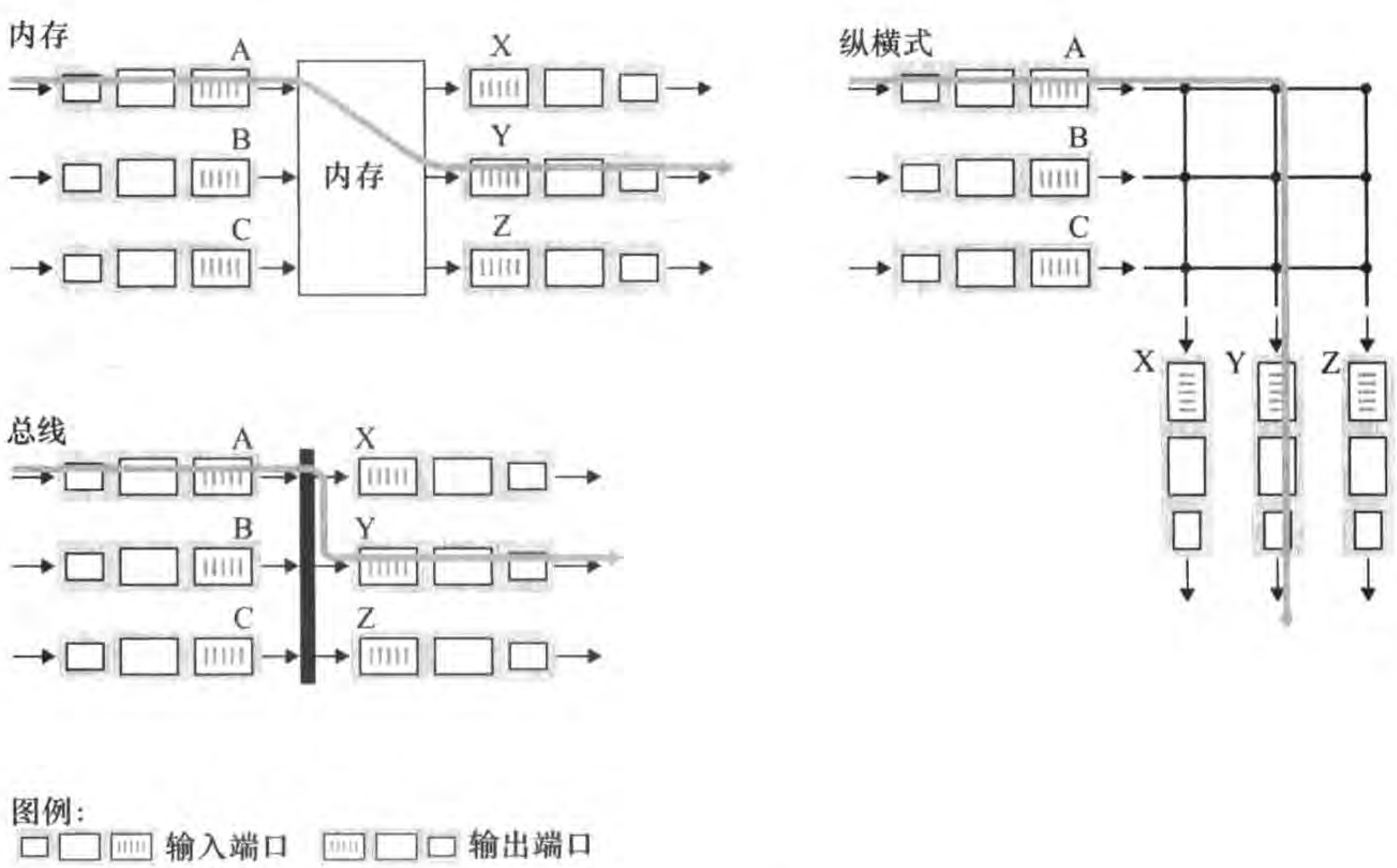


图4-6　三种交换技术

- 经内存交换。最简单、最早的路由器是传统的计算机，在输入端口与输出端口之间的交换是在CPU（路由选择处理器）的直接控制下完成的。输入与输出端口的功能就像在传统操作系统中的I/O设备一样。一个分组到达一个输入端口时，该端口会先通过中断方式向路由选择处理器发出信号。于是，该分组从输入端口处被复制到处理器内存中。路由选择处理器则从其首部中提取目的地址，在转发表中找出适当的输出端口，并将该分组复制到输出端口的缓存中。在这种情况下，如果内存带宽为每秒可写进内存或从内存读出最多 B 个分组，则总的转发吞吐量（分组从输入端口被传送到输出端口的总速率）必然小于 $B/2$。也要注意到不能同时转发两个分组，即使它们有不同的目的端口，因为经过共享系统总线一次仅能执行一个内存读/写。

  许多现代路由器通过内存进行交换。然而，与早期路由器的一个主要差别是，目的地址的查找和将分组存储（交换）进适当的内存存储位置是由输入线路卡来处理的。在某些方面，经内存交换的路由器看起来很像共享内存的多处理器，用一个线路卡上的处理将分组交换（写）进适当的输出端口的内存中。Cisco的Catalyst 8500系列的交换机［Cisco 8500 2016］是经共享内存转发分组的。
- 经总线交换。在这种方法中，输入端口经一根共享总线将分组直接传送到输出端口，不需要路由选择处理器的干预。通常按以下方式完成该任务：让输入端口为分组预先计划一个交换机内部标签（首部），指示本地输出端口，使分组在总线上传送和传输到输出端口。该分组能由所有输出端口收到，但只有与该标签匹配的端口才能保存该分组。然后标签在输出端口被去除，因为其仅用于交换机内部来跨越总线。如果多个分组同时到达路由器，每个位于不同的输出端口，除了一个分组外所有其他分组必须等待，因为一次只有一个分组能够跨越总线。因为每个分组必须跨过单一总线，故路由器的交换带宽受总线速率的限制；在环状交叉路的类比中，这相当于环状交叉路一次仅包含一辆汽车。尽管如此，对于运行在小型局域网和企业网中的路由器来说，通过总线交换通常足够用了。Cisco 6500路由器［Cisco 6500 2016］内部通过一个32Gbps背板总线来交换分组。
- 经互联网络交换。克服单一、共享式总线带宽限制的一种方法是，使用一个更复杂的互联网络，例如过去在多处理器计算机体系结构中用来互联多个处理器的网络。纵横式交换机就是一种由 $2N$ 条总线组成的互联网络，它连接 N 个输入端口与 N 个输出端口，如图4-6所示。每条垂直的总线在交叉点与每条水平的总线交叉，交叉点通过交换结构控制器（其逻辑是交换结构自身的一部分）能够在任何时候开启和闭合。当某分组到达端口A，需要转发到端口Y时，交换机控制器闭合总线A和Y交叉部位的交叉点，然后端口A在其总线上发送该分组，该分组仅由总线Y接收。注意到来自端口B的一个分组在同一时间能够转发到端口X，因为A到Y和B到X的分组使用不同的输入和输出总线。因此，与前面两种交换方法不同，纵横式网络能够并行转发多个分组。纵横式交换机是**非阻塞的**（non-blocking），即只要没有其他分组当前被转发到该输出端口，转发到输出端口的分组将不会被到达输出端口的分组阻塞。然而，如果来自两个不同输入端口的两个分组其目的地为相同的输出端口，则一个分组必须在输入端等待，因为在某个时刻经给定总线仅能够发送一个分组。Cisco 12000系列交换机［Cisco 12000 2016］使用了一个互联网络；Cisco 7600系列能被配置为使用总线或者纵横式交换机

[Cisco 7600 2016]。

更为复杂的互联网络使用多级交换元素，以使来自不同输入端口的分组通过交换结构同时朝着相同的输出端口前行。对交换机体系结构的展望可参见[Tobagi 1990]。Cisco CRS 利用了一种三级非阻塞交换策略。路由器的交换能力也能够通过并行运行多种交换结构进行扩展。在这种方法中，输入端口和输出端口被连接到并行运行的 N 个交换结构。一个输入端口将一个分组分成 K 个较小的块，并且通过 N 个交换结构中的 K 个发送（“喷射”）这些块到所选择的输出端口，输出端口再将 K 个块装配还原成初始的分组。

4.2.3 输出端口处理

如图 4-7 中所示，输出端口处理取出已经存放在输出端口内存中的分组并将其发送到输出链路上。这包括选择和取出排队的分组进行传输，执行所需的链路层和物理层传输功能。

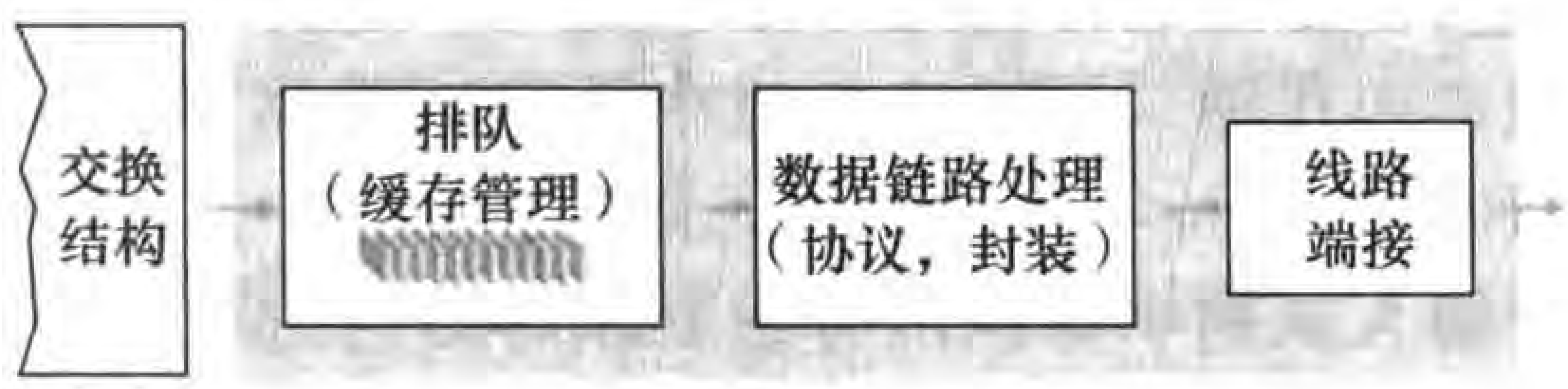


图 4-7　输出端口处理

4.2.4 何处出现排队

如果我们考虑显示在图 4-6 中的输入和输出端口功能及其配置，下列情况是一目了然的：在输入端口和输出端口处都可以形成分组队列，就像在环状交叉路的类比中我们讨论过的情况，即汽车可能等待在流量交叉点的入口和出口。排队的位置和程度（或者在输入端口排队，或者在输出端口排队）将取决于流量负载、交换结构的相对速率和线路速率。我们现在更为详细一点考虑这些队列，因为随着这些队列的增长，路由器的缓存空间最终将会耗尽，并且当无内存可用于存储到达的分组时将会出现**丢包**（packet loss）。回想前面的讨论，我们说过分组“在网络中丢失”或“被路由器丢弃”。正是在一台路由器的这些队列中，这些分组被实际丢弃或丢失。

假定输入线路速度与输出线路速度（传输速率）是相同的，均为 R_{line}（单位为每秒分组数），并且有 N 个输入端口和 N 个输出端口。为进一步简化讨论，假设所有分组具有相同的固定长度，分组以同步的方式到达输入端口。这就是说，在任何链路发送分组的时间等于在任何链路接收分组的时间，在这样的时间间隔内，在一个输入链路上能够到达 0 个或 1 个分组。定义交换结构传送速率 R_{switch} 为从输入端口到输出端口能够移动分组的速率。如果 R_{switch} 比 R_{line} 快 N 倍，则在输入端口处仅会出现微不足道的排队。这是因为即使在最坏情况下，所有 N 条输入线路都在接收分组，并且所有的分组将被转发到相同的输出端口，每批 N 个分组（每个输入端口一个分组）也能够在下一批到达前通过交换结构处理完毕。

1. 输入排队

如果交换结构不能快得（相对于输入线路速度而言）使所有到达分组无时延地通过它传送，会发生什么情况呢？在这种情况下，在输入端口也将出现分组排队，因为到达的分组必须加入输入端口队列中，以等待通过交换结构传送到输出端口。为了举例说明这种排队的重要后果，考虑纵横式交换结构，并假定：①所有链路速度相同；②一个分组能够以一条输入链路接收一个分组所用的相同的时间量，从任意一个输入端口传送到给定的输出

端口；③分组按 FCFS 方式，从一指定输入队列移动到其要求的输出队列中。只要其输出端口不同，多个分组可以被并行传送。然而，如果位于两个输入队列前端的两个分组是发往同一输出队列的，则其中的一个分组将被阻塞，且必须在输入队列中等待，因为交换结构一次只能传送一个分组到某指定端口。

图4-8 显示了一个例子，其中在输入队列前端的两个分组（带深色阴影）要发往同一个右上角输出端口。假定该交换结构决定发送左上角队列前端的分组。在这种情况下，左下角队列中的深色阴影分组必须等待。但不仅该分组要等待，左下角队列中排在该分组后面的浅色阴影分组也要等待，即使右中侧输出端口（浅色阴影分组的目的地）中无竞争。这种现象叫作输入排队交换机中的**线路前部**（Head-Of-the-Line，HOL）**阻塞**，即在一个输入队列中排队的分组必须等待通过交换结构发送（即使输出端口是空闲的），因为它被位于线路前部的另一个分组所阻塞。[Karol 1987] 指出，由于 HOL 阻塞，只要输入链路上的分组到达速率达到其容量的 58%，在某些假设前提下，输入队列长度就将无限制地增大（不严格地讲，这等同于说将出现大量的丢包）。[McKeown 1997b] 讨论了多种解决 HOL 阻塞的方法。

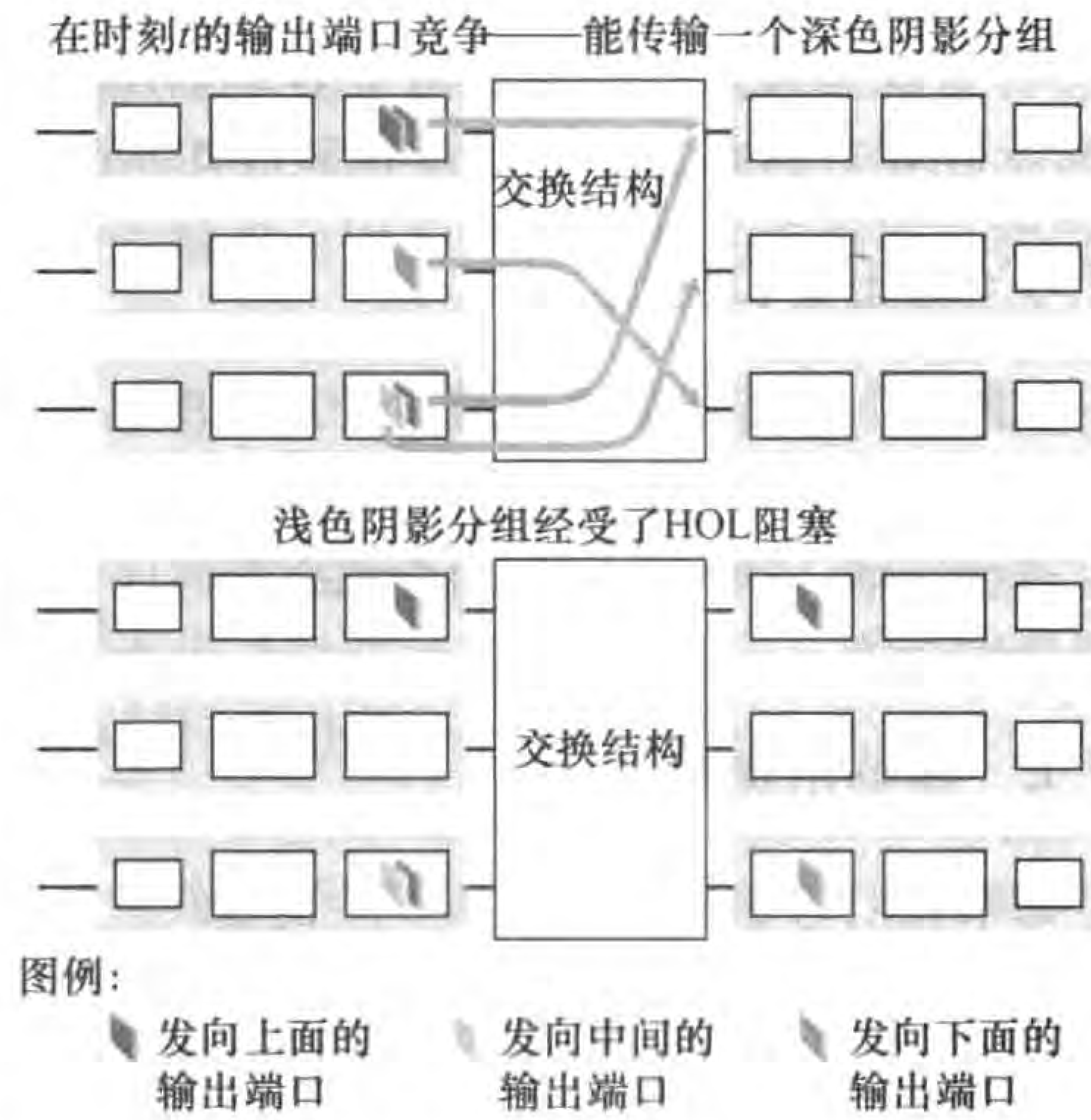


图 4-8 在一个输入排队交换机中的 HOL 阻塞

2. **输出排队**

我们接下来考虑在交换机的输出端口是否会出现排队。再次假定 R_{switch} 比 R_{line} 快 N 倍，并且到达 N 个输入端口的每个端口的分组，其目的地是相同的输出端口。在这种情况下，在向输出链路发送一个分组的时间内，将有 N 个新分组到达该输出端口（N 个输入端口的每个都到达 1 个）。因为输出端口在一个单位时间（该分组的传输时间）内仅能传输一个分组，这 N 个到达分组必须排队（等待）经输出链路传输。在正好传输 N 个分组（这些分组是前面正在排队的）之一的时间中，可能又到达 N 个分组，等等。所以，分组队列能够在输出端口形成，即使交换结构比端口线路速率快 N 倍。最终，排队的分组数量能够变得足够大，耗尽输出端口的可用内存。

当没有足够的内存来缓存一个入分组时，就必须做出决定：要么丢弃到达的分组（采用一种称为**弃尾**（drop-tail）的策略），要么删除一个或多个已排队的分组为新来的分组腾出空间。在某些情况下，在缓存填满之前便丢弃一个分组（或在其首部加上标记）的做法是有利的，这可以向发送方提供一个拥塞信号。已经提出和分析了许多分组丢弃与标记策略 [Labrador 1999，Hollot 2002]，这些策略统称为**主动队列管理**（Active Queue Management，AQM）算法。**随机早期检测**（Random Early Detection，RED）算法是得到最广泛研究和实现的 AQM 算法之一 [Christiansen 2001；Floyd 2016]。

在图4-9中图示了输出端口的排队情况。在时刻 t，每个入端输入端口都到达了一个分组，每个分组都是发往最上侧的输出端口。假定线路速度相同，交换机以3倍于线路速度的速度运行，一个时间单位（即接收或发送一个分组所需的时间）以后，所有三个初始分组都被传送到输出端口，并排队等待传输。在下一个时间单位中，这三个分组中的一个将通过输出链路发送出去。在这个例子中，又有两个新分组已到达交换机的入端；这些分组之一要发往最上侧的输出端口。这样的后果是，输出端口的**分组调度**（packet scheduler）在这些排队分组中选择一个分组来传输，这就是我们将在下节中讨论的主题。

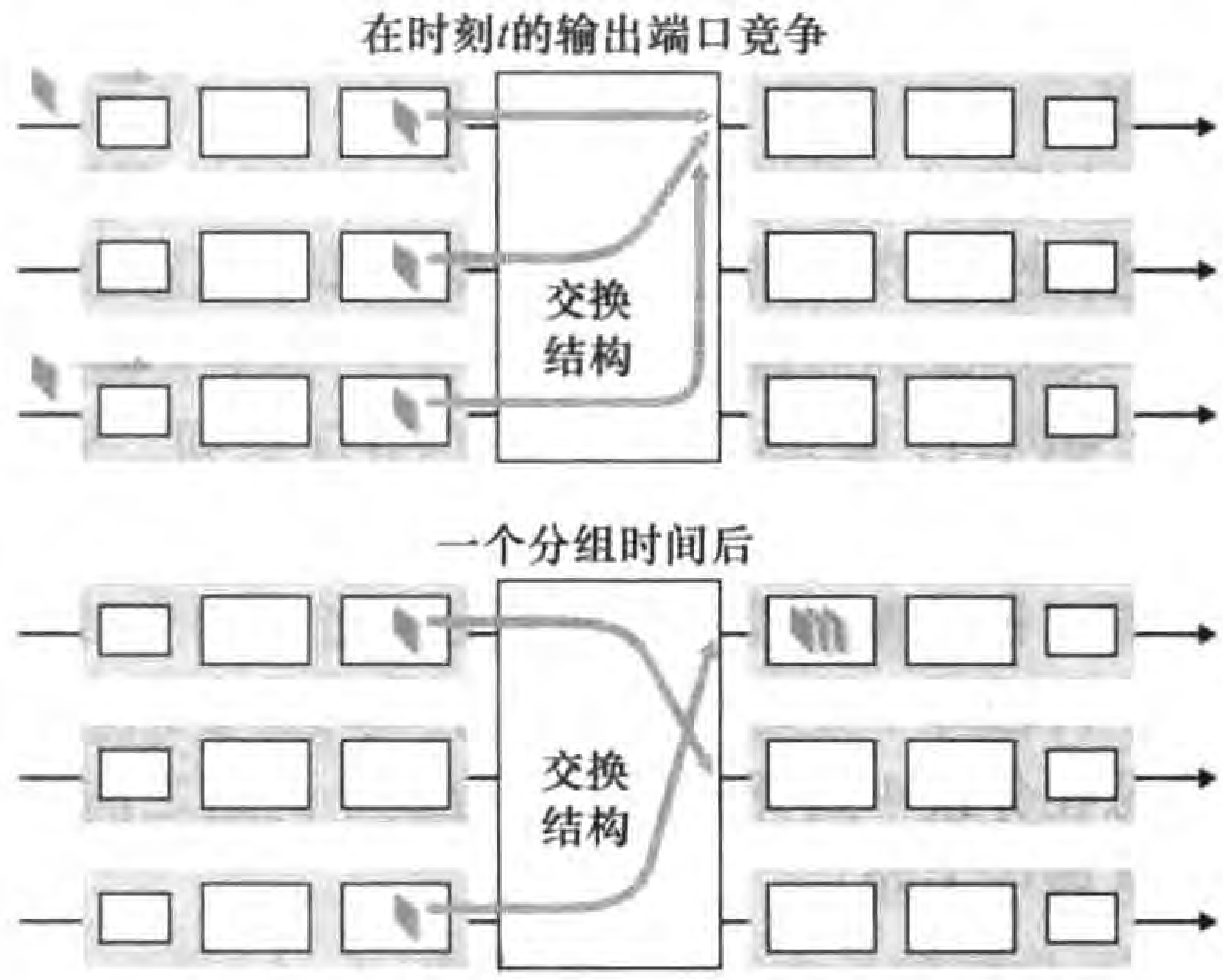


图4-9 输出端口排队

假定需要路由器缓存来吸收流量负载的波动，一个自然而然的问题就是需要多少缓存。多年以来，用于缓存长度的经验方法是［RFC 3439］，缓存数量（B）应当等于平均往返时延（RTT，比如说250ms）乘以链路的容量（C）。这个结果是基于相对少量的TCP流的排队动态性分析得到的［Villamizar 1994］。因此，一条具有250ms RTT的10Gbps链路需要的缓存量等于 $B = \text{RTT} \cdot C = 2.5\text{Gb}$。然而，最近的理论和试验研究［Appenzeller 2004］表明，当有大量的TCP流（N 条）流过一条链路时，缓存所需要的数量是 $B = \text{RTT} \cdot C/\sqrt{N}$。对于通常有大量流经过的大型主干路由器链路（参见如［Fraleigh 2003］），N 的值可能非常大，所需的缓存长度的减小相当明显。［Appenzeller 2004；Wischik 2005；Beheshti 2008］从理论、实现和运行的角度提供了可读性很强的有关缓存长度问题的讨论。

4.2.5 分组调度

现在我们转而讨论确定次序的问题，即排队的分组如何经输出链路传输的问题。以前你自己无疑在许多场合都排长队等待过，并观察过等待的客户怎样被服务，你无疑也熟悉路由器中常用的许多排队规则。有一种是先来先服务（FCFS，也称之为先进先出（FIFO））。这是英国人人共知的规则，用于病人就诊、公交车站和市场中的有序FCFS队列。（哦，你排队了吗？）有些国家基于优先权运转，即给一类等待客户超越其他等待客户的优先权服务。也有循环排队，其中客户也被划分为类别（与在优先权队列一样），但每类用户依次序提供服务。

1. 先进先出

图4-10显示了对于**先进先出**（First-In-First-Out，FIFO）链路调度规则的排队模型的抽象。如果链路当前正忙于传输另一个分组，到达链路输出队列的分组要排队等待传输。如果没有足够的缓存空间来容纳到达的分组，队列的分组丢弃策略则确定该分组是否将被丢弃（丢失）或者

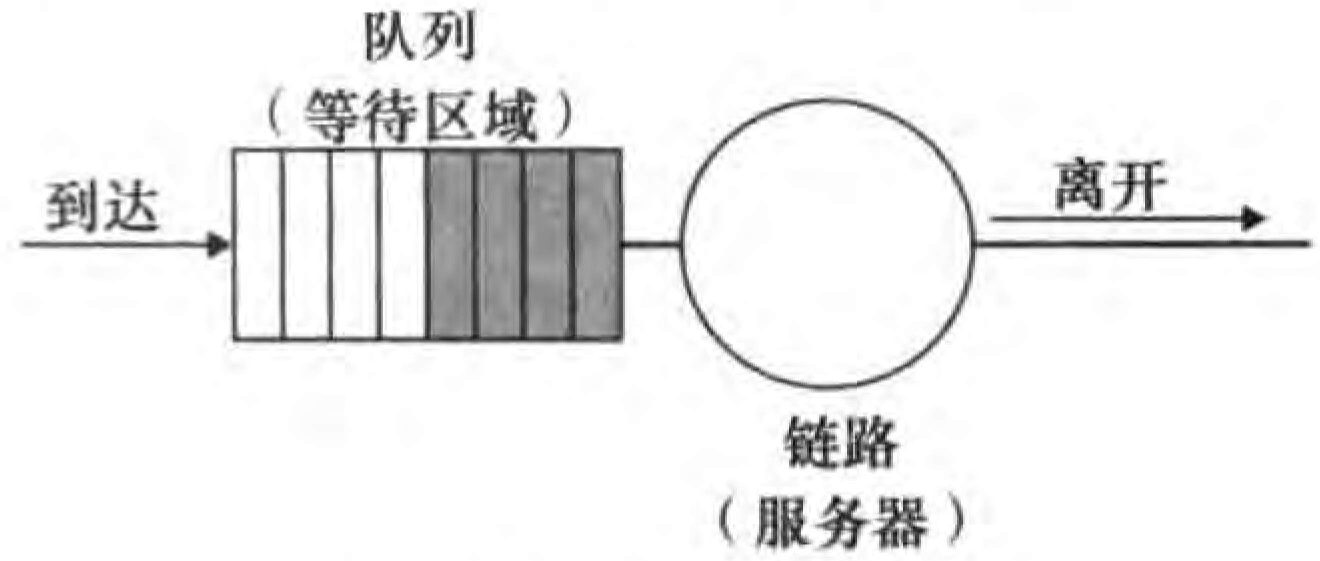


图4-10 FIFO排队抽象

从队列中去除其他分组以便为到达的分组腾出空间，如前所述。在下面的讨论中，我们将忽视分组丢弃。当一个分组通过输出链路完全传输（也就是接收服务）时，从队列中去除它。

FIFO（也称为先来先服务，FCFS）调度规则按照分组到达输出链路队列的相同次序来选择分组在链路上传输。我们都很熟悉服务中心的 FIFO 排队，在那里到达的顾客加入单一等待队列的最后，保持次序，然后当他们到达队伍的前面时就接受服务。

图 4-11 显示了运行中的 FIFO 队列。分组的到达由上部时间线上带编号的箭头来指示，用编号指示了分组到达的次序。各个分组的离开表示在下部时间线的下面。分组在服务中（被传输）花费的时间是通过这两个时间线之间的阴影矩形来指示的。假定在这个例子中传输每个分组用去 3 个单位时间。利用 FIFO 规则，分组按照到达的相同次序离开。注意在分组 4 离开之后，在分组 5 到达之前链路保持空闲（因为分组 1 ~4 已经被传输并从队列中去除）。

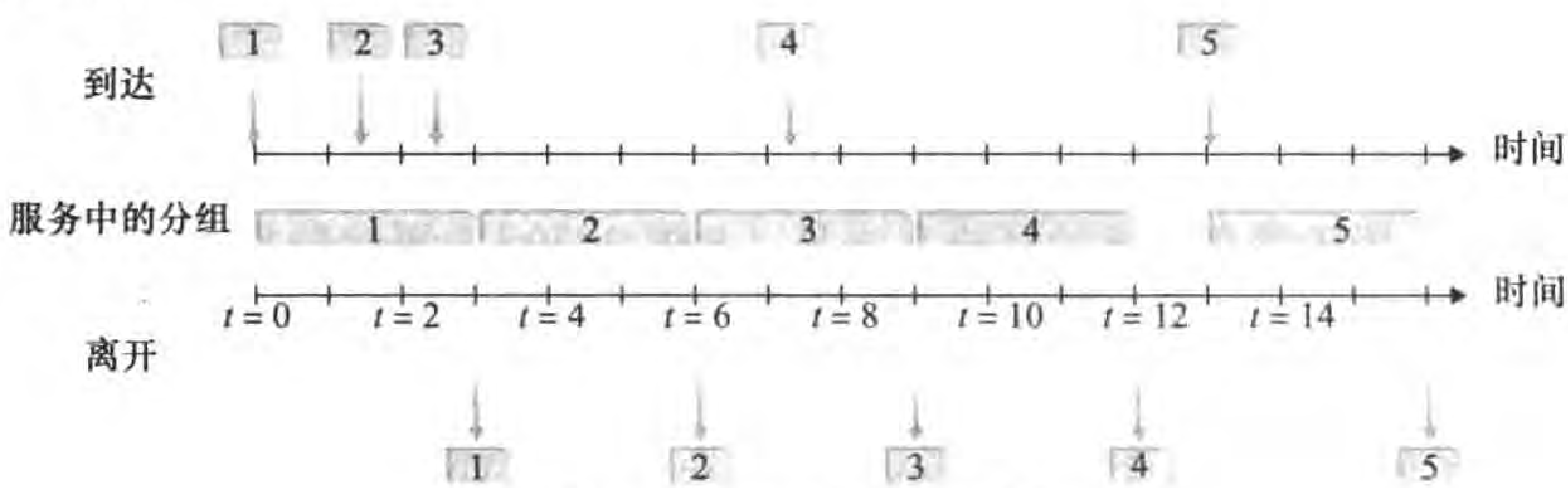


图 4-11　运行中的 FIFO 队列

2. 优先权排队

在**优先权排队**（priority queuing）规则下，到达输出链路的分组被分类放入输出队列中的优先权类，如图 4-12 所示。在实践中，网络操作员可以配置一个队列，这样携带网络管理信息的分组（例如，由源或目的 TCP/UDP 端口号所标识）获得超过用户流量的优先权；此外，基于 IP 的实时话音分组可能获得超过非实时流量（如 SMTP 或 IMAP 电子邮件分组）的优先权。每个优先权类通常都有自己的队列。当选择一个分组传输时，优先权排队规则将从队列为非空（也就是有分组等待传输）的最高优先权类中传输一个分组。在同一优先权类的分组之间的选择通常以 FIFO 方式完成。

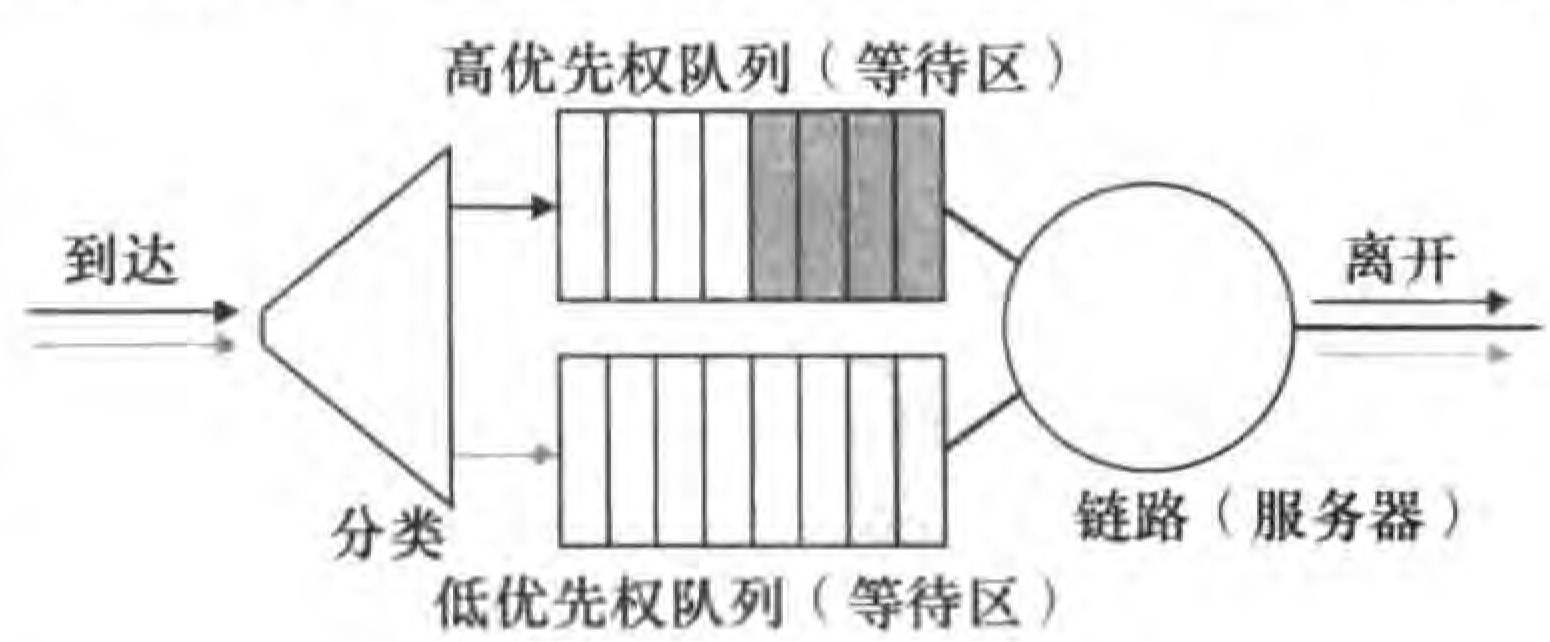


图 4-12　优先权排队模型

图 4-13 描述了有两个优先权类的一个优先权队列的操作。分组 1、3 和 4 属于高优先权类，分组 2 和 5 属于低优先权类。分组 1 到达并发现链路是空闲的，就开始传输。在分组 1 的传输过程中，分组 2 和 3 到达，并分别在低优先权和高优先权队列中排队。在传输完分组 1 后，分组 3（一个高优先权的分组）被选择在分组 2（尽管它到达得较早，但它是一个低优先权分组）之前传输。在分组 3 的传输结束后，分组 2 开始传输。分组 4（一个高优先权分组）在分组 2（一个低优先权分组）的传输过程中到达。在**非抢占式优先权排队**（non-preemptive priority queuing）规则下，一旦分组

开始传输，就不能打断。在这种情况下，分组4排队等待传输，并在分组2传输完成之后开始传输。

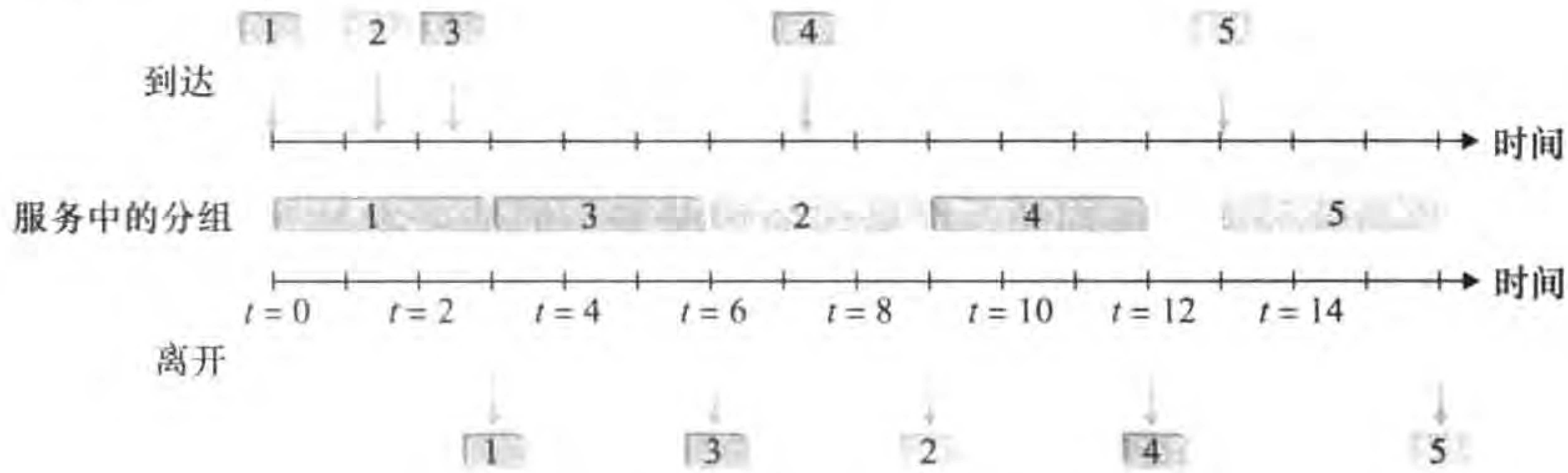


图4-13 优先权队列的操作

3. 循环和加权公平排队

在**循环排队规则**（round robin queuing discipline）下，分组像使用优先权排队那样被分类。然而，在类之间不存在严格的服务优先权，循环调度器在这些类之间轮流提供服务。在最简单形式的循环调度中，类1的分组被传输，接着是类2的分组，接着又是类1的分组，再接着又是类2的分组，等等。一个所谓的**保持工作排队**（work-conserving queuing）规则在有（任何类的）分组排队等待传输时，不允许链路保持空闲。当寻找给定类的分组但是没有找到时，保持工作的循环规则将立即检查循环序列中的下一个类。

图4-14描述了一个两类循环队列的操作。在这个例子中，分组1、2和4属于第一类，分组3和5属于第二类。分组1一到达输出队列就立即开始传输。分组2和3在分组1的传输过程中到达，因此排队等待传输。在分组1传输后，链路调度器查找类2的分组，因此传输分组3。在分组3传输完成后，调度器查找类1的分组，因此传输分组2。在分组2传输完成后，分组4是唯一排队的分组，因此在分组2后立刻传输分组4。

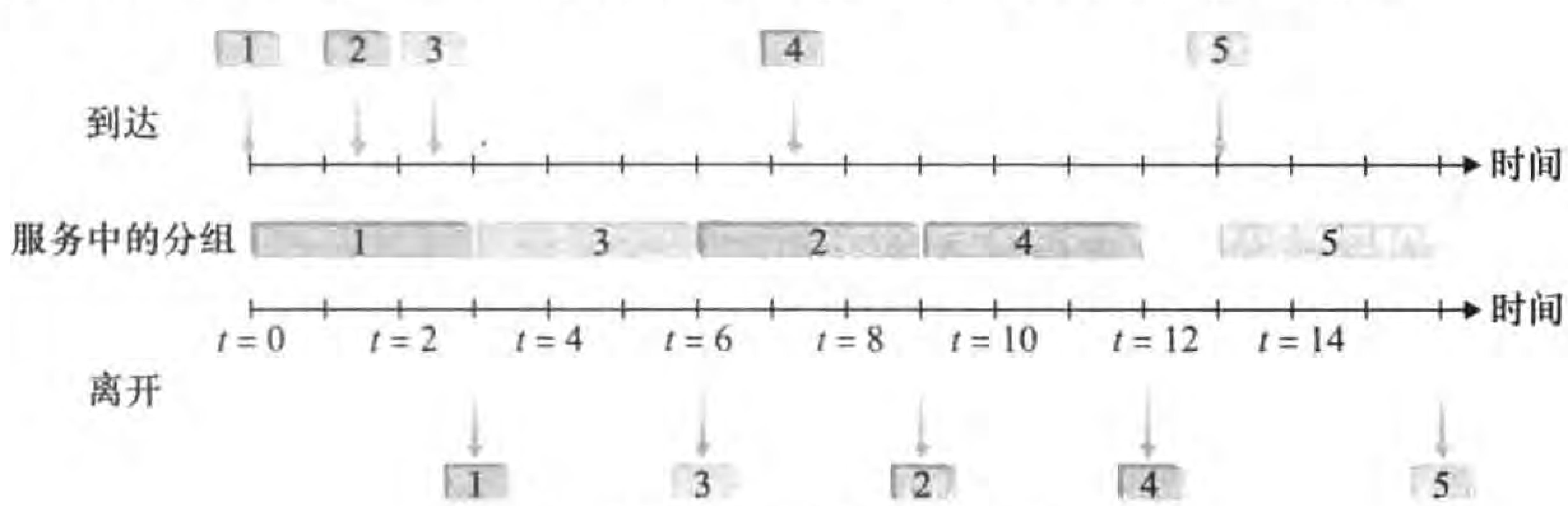


图4-14 两类循环队列的操作

一种通用形式的循环排队已经广泛地实现在路由器中，它就是所谓的**加权公平排队**（Weighted Fair Queuing，WFQ）规则［Demers 1990；Parekh 1993；Cisco QoS 2016］。图4-15对WFQ进行了描述。其中，到达的分组被分类并在合适的每个类的等待区域排队。与使用循环调度一样，WFQ调度器也以循环的方式为各个类提供服务，即首先服务第1类，然后服务第2类，接着再服务第3类，然后

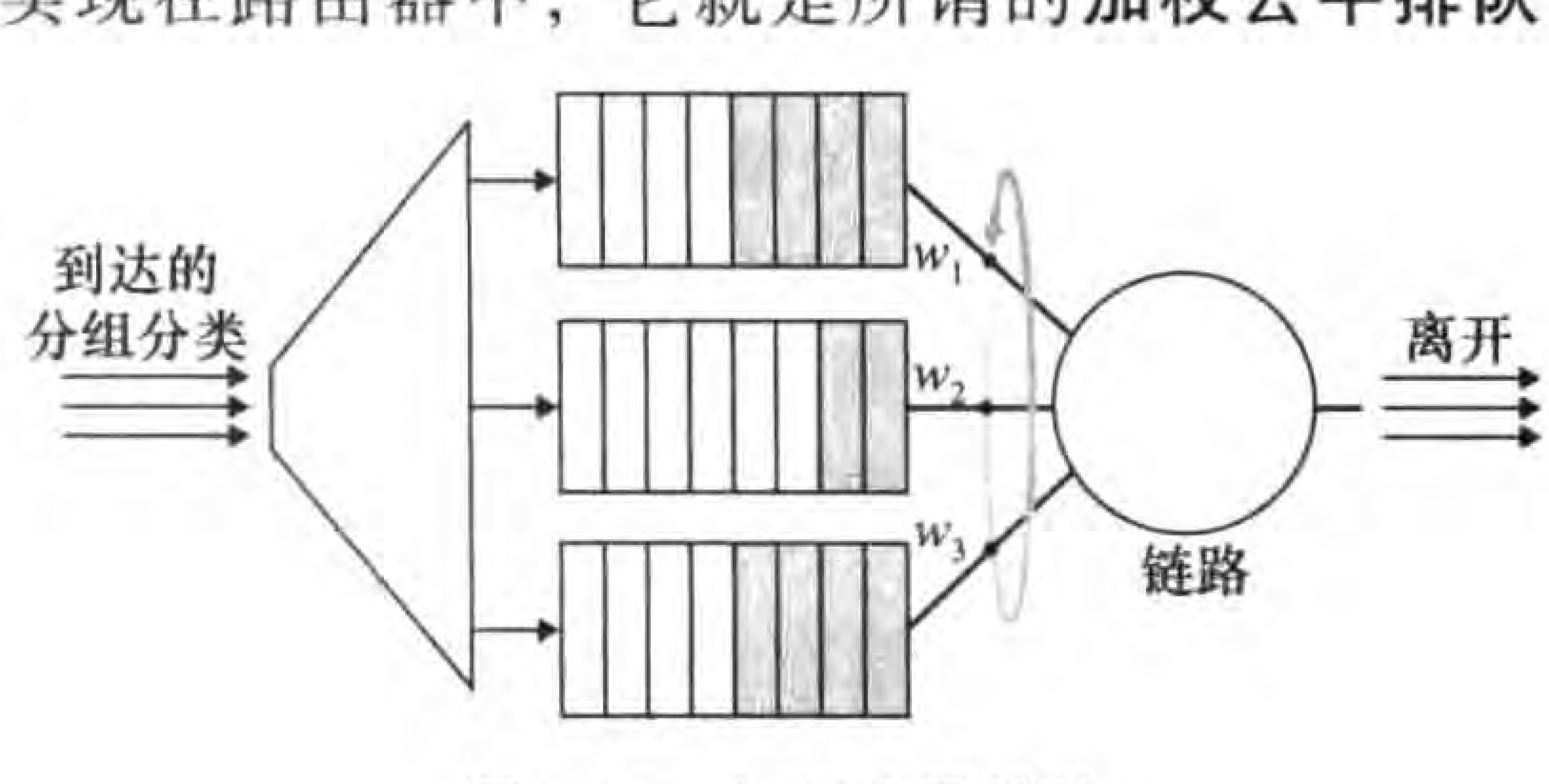


图4-15 加权公平排队

（假设有3个类别）重复这种服务模式。WFQ也是一种保持工作排队规则，因此在发现一个空的类队列时，它立即移向服务序列中的下一个类。

WFQ和循环排队的不同之处在于，每个类在任何时间间隔内可能收到不同数量的服务。具体而言，每个类 i 被分配一个权 w_i。使用WFQ方式，在类 i 有分组要发送的任何时间间隔中，第 i 类将确保接收到的服务部分等于 $w_i/(\sum w_j)$，式中分母中的和是计算所有有分组排队等待传输的类别得到的。在最坏的情况下，即使所有的类都有分组排队，第 i 类仍然保证分配到带宽的 $w_i/(\sum w_j)$ 部分。因此，对于一条传输速率为 R 的链路，第 i 类总能获得至少为 $R \cdot w_i/(\sum w_j)$ 的吞吐量。我们对WFQ的描述理想化了，因为没有考虑这样的事实：分组是离散的数据单元，并且不能打断一个分组的传输来开始传输另一个分组；[Demers 1990；Parekh 1993] 讨论了这个分组化问题。

4.3　网际协议：IPv4、寻址、IPv6及其他

到目前为止，我们在第4章中对网络层的学习，包括网络层的数据平面和控制平面组件概念，转发和路由选择之间的区别，各种网络服务模型的标识和对路由器内部的观察，并未提及任何特定的计算机网络体系结构或协议。在这节中，我们将关注点转向今天的因特网网络层的关键方面和著名的网际协议（IP）。

今天有两个版本的IP正在使用。在4.3.1节中，我们首先研究广泛部署的IP版本4，这通常简单地称为IPv4 [RFC 791]。在4.3.5节中，我们将仔细考察IP版本6 [RFC 2460；RFC 4291]，它已经被提议替代IPv4。在中间，我们将主要学习因特网编址，这是一个看起来相当枯燥和面向细节的主题，但是这对理解因特网网络层如何工作是至关重要的。掌握IP编址就是掌握因特网的网络层！

4.3.1　IPv4数据报格式

前面讲过网络层分组被称为*数据报*。我们以概述IPv4数据报的语法和语义开始对IP的学习。你也许认为没有什么比一个分组的比特的语法和语义更加枯燥无味的了。无论如何，数据报在因特网中起着重要作用，每个网络行业的学生和专业人员都需要理解它、吸收它并掌握它（只是理解协议首部的确能够使学习成为有趣的事，请查阅 [Pomeranz 2010]）。IPv4数据报格式如图4-16所示。

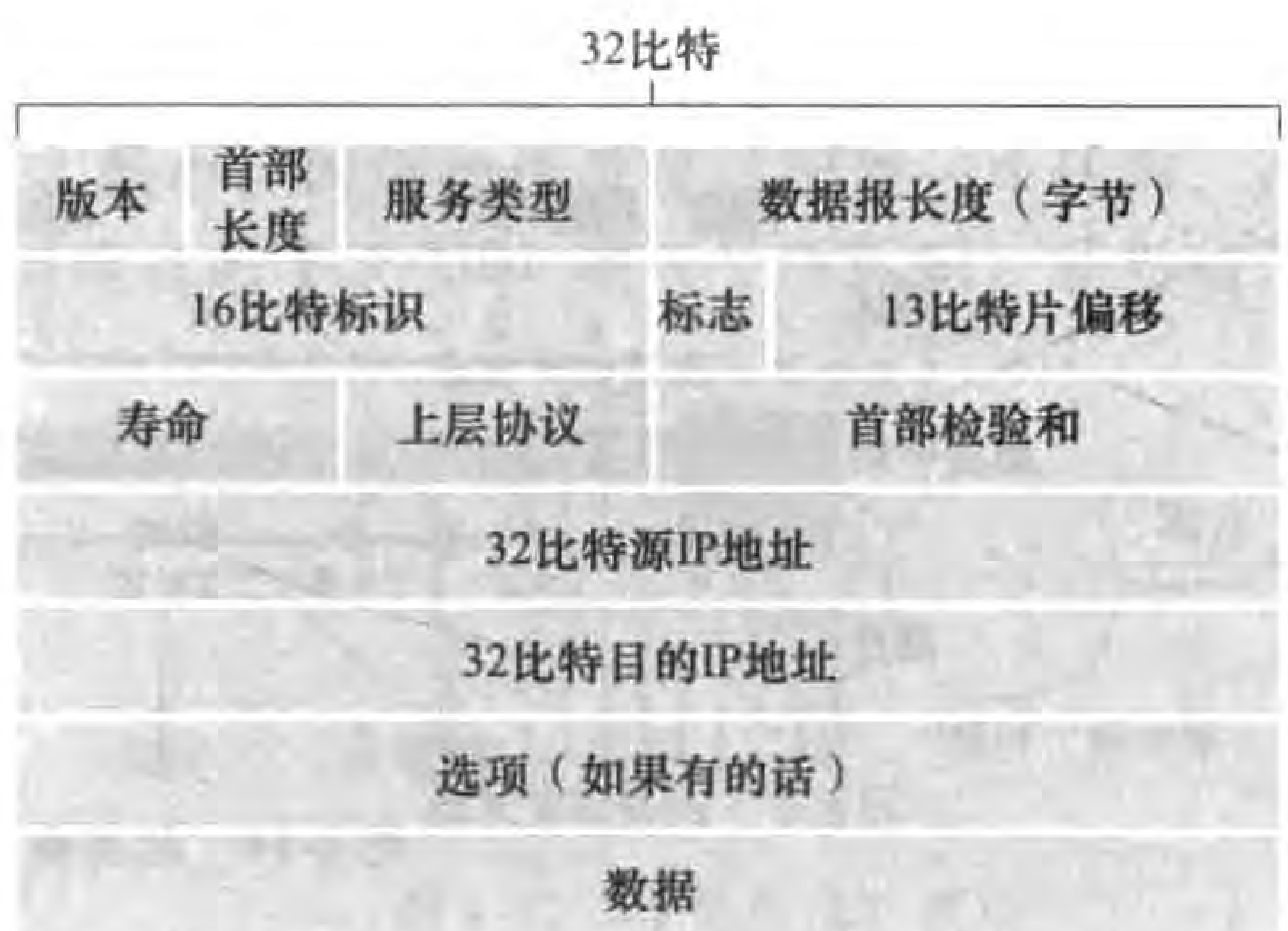


图4-16　IPv4数据报格式

IPv4数据报中的关键字段如下：

- *版本（号）*。这4比特规定了数据报的IP协议版本。通过查看版本号，路由器能够确定如何解释IP数据报的剩余部分。不同的IP版本使用不同的数据报格式。IPv4的数据报格式如图4-16所示。新版本的IP（IPv6）的数据报格式将在4.3.5节中讨论。
- *首部长度*。因为一个IPv4数据报可包含一些可变数量的选项（这些选项包括在IPv4数据报首部中），故需要用这4比特来确定IP数据报中载荷（例如在这个数

据报中被封装的运输层报文段）实际开始的地方。大多数 IP 数据报不包含选项，所以一般的 IP 数据报具有 20 字节的首部。

- 服务类型。服务类型（TOS）比特包含在 IPv4 首部中，以便使不同类型的 IP 数据报（例如，一些特别要求低时延、高吞吐量或可靠性的数据报）能相互区别开来。例如，将实时数据报（如用于 IP 电话应用）与非实时流量（如 FTP）区分开也许是有用的。提供特定等级的服务是一个由网络管理员对路由器确定和配置的策略问题。我们在 3.7.2 节讨论明确拥塞通告所使用的两个 TOS 比特时也学习过。
- 数据报长度。这是 IP 数据报的总长度（首部加上数据），以字节计。因为该字段长为 16 比特，所以 IP 数据报的理论最大长度为 65 535 字节。然而，数据报很少有超过 1500 字节的，该长度使得 IP 数据报能容纳最大长度以太网帧的载荷字段。
- 标识、标志、片偏移。这三个字段与所谓 IP 分片有关，这是一个我们将很快要考虑的主题。有趣的是，新版本的 IP（即 IPv6）不允许在路由器上对分组分片。
- 寿命。寿命（Time-To-Live，TTL）字段用来确保数据报不会永远（如由于长时间的路由选择环路）在网络中循环。每当一台路由器处理数据报时，该字段的值减 1。若TTL字段减为 0，则该数据报必须丢弃。
- 协议。该字段通常仅当一个 IP 数据报到达其最终目的地时才会有用。该字段值指示了 IP 数据报的数据部分应交给哪个特定的运输层协议。例如，值为 6 表明数据部分要交给 TCP，而值为 17 表明数据要交给 UDP。对于所有可能值的列表，参见［IANA Protocol Numbers 2016］。注意在 IP 数据报中的协议号所起的作用，类似于运输层报文段中端口号字段所起的作用。协议号是将网络层与运输层绑定到一起的黏合剂，而端口号是将运输层和应用层绑定到一起的黏合剂。我们将在第 6 章看到，链路层帧也有一个特殊字段用于将链路层与网络层绑定到一起。
- 首部检验和。首部检验和用于帮助路由器检测收到的 IP 数据报中的比特错误。首部检验和是这样计算的：将首部中的每 2 个字节当作一个数，用反码算术对这些数求和。如在 3.3 节讨论的那样，该和的反码（被称为因特网检验和）存放在检验和字段中。路由器要对每个收到的 IP 数据报计算其首部检验和，如果数据报首部中携带的检验和与计算得到的检验和不一致，则检测出是个差错。路由器一般会丢弃检测出错误的数据报。注意到在每台路由器上必须重新计算检验和并再次存放到原处，因为TTL字段以及可能的选项字段会改变。关于计算因特网检验和的快速算法的有趣讨论参见［RFC 1071］。此时，一个经常问的问题是：为什么 TCP/IP 在运输层与网络层都执行差错检测？这种重复检测有几种原因。首先，注意到在 IP 层只对 IP 首部计算了检验和，而 TCP/UDP 检验和是对整个 TCP/UDP 报文段进行的。其次，TCP/UDP 与 IP 不一定都必须属于同一个协议栈。原则上，TCP 能够运行在一个不同的协议（如 ATM）上［Black 1995］，而 IP 能够携带不一定要传递给 TCP/UDP 的数据。
- 源和目的 IP 地址。当某源生成一个数据报时，它在源 IP 字段中插入它的 IP 地址，在目的 IP 地址字段中插入其最终目的地的地址。通常源主机通过 DNS 查找来决定目的地址，如在第 2 章中讨论的那样。我们将在 4.3.3 节中详细讨论 IP

编址。

- **选项**。选项字段允许 IP 首部被扩展。首部选项意味着很少使用，因此决定对每个数据报首部不包括选项字段中的信息，这样能够节约开销。然而，少量选项的存在的确使问题复杂了，因为数据报首部长度可变，故不能预先确定数据字段从何处开始。而且还因为有些数据报要求处理选项，而有些数据报则不要求，故导致一台路由器处理一个 IP 数据报所需的时间变化可能很大。这些考虑对于高性能路由器和主机上的 IP 处理来说特别重要。由于这样或那样的原因，在 IPv6 首部中已去掉了 IP 选项，如 4.3.5 节中讨论的那样。
- **数据（有效载荷）**。我们来看看最后也是最重要的字段，这是数据报存在的*首要理由*！在大多数情况下，IP 数据报中的数据字段包含要交付给目的地的运输层报文段（TCP 或 UDP）。然而，该数据字段也可承载其他类型的数据，如 ICMP 报文（在 5.6 节中讨论）。

注意到一个 IP 数据报有总长为 20 字节的首部（假设无选项）。如果数据报承载一个 TCP 报文段，则每个（无分片的）数据报共承载了总长 40 字节的首部（20 字节的 IP 首部加上 20 字节的 TCP 首部）以及应用层报文。

4.3.2　IPv4 数据报分片

在第 6 章中我们将看到，并不是所有链路层协议都能承载相同长度的网络层分组。有的协议能承载大数据报，而有的协议只能承载小分组。例如，以太网帧能够承载不超过 1500 字节的数据，而某些广域网链路的帧可承载不超过 576 字节的数据。一个链路层帧能承载的最大数据量叫作**最大传送单元**（Maximum Transmission Unit，MTU）。因为每个 IP 数据报封装在链路层帧中从一台路由器传输到下一台路由器，故链路层协议的 MTU 严格地限制着 IP 数据报的长度。对 IP 数据报长度具有严格限制并不是主要问题。问题在于在发送方与目的地路径上的每段链路可能使用不同的链路层协议，且每种协议可能具有不同的 MTU。

为了更好地理解这一转发问题，想象你是一台互联几条链路的路由器，且每条链路运行具有不同 MTU 的链路层协议。假定你从某条链路收到一个 IP 数据报，通过检查转发表确定出链路，并且该条出链路的 MTU 比该 IP 数据报的长度要小。此时你会感到慌乱，如何将这个过大的 IP 分组挤进链路层帧的有效载荷字段呢？解决该问题的方法是将 IP 数据报中的数据分片成两个或更多个较小的 IP 数据报，用单独的链路层帧封装这些较小的 IP 数据报，然后通过输出链路发送这些帧。每个这些较小的数据报都称为**片**（fragment）。

片在其到达目的地运输层以前需要重新组装。TCP 与 UDP 的确都希望从网络层收到完整的、未分片的报文。IPv4 的设计者感到在路由器中重新组装数据报会给协议带来相当大的复杂性并且影响路由器的性能。（如果你是一台路由器，你愿意将重新组装报文片放在你必须要做的各种各样工作的首位吗？）为坚持网络内核保持简单的原则，IPv4 的设计者决定将数据报的重新组装工作放到端系统中，而不是放到网络路由器中。

当一台目的主机从相同源收到一系列数据报时，它需要确定这些数据报中的某些是否是一些原来较大的数据报的片。如果某些数据报是这些片的话，则它必须进一步确定何时收到了最后一片，并且如何将这些接收到的片拼接到一起以形成初始的数据报。为了让目

的主机执行这些重新组装任务，IPv4 的设计者将标识、标志和片偏移字段放在 IP 数据报首部中。当生成一个数据报时，发送主机在为该数据报设置源和目的地址的同时贴上标识号。发送主机通常将它发送的每个数据报的标识号加 1。当某路由器需要对一个数据报分片时，形成的每个数据报（即片）具有初始数据报的源地址、目的地址与标识号。当目的地从同一发送主机收到一系列数据报时，它能够检查数据报的标识号以确定哪些数据报实际上是同一较大数据报的片。由于 IP 是一种不可靠的服务，一个或多个片可能永远到达不了目的地。因为这种原因，为了让目的主机绝对地相信它已收到了初始数据报的最后一个片，最后一个片的标志比特被设为 0，而所有其他片的标志比特被设为 1。另外，为了让目的主机确定是否丢失了一个片（且能按正确的顺序重新组装片），使用偏移字段指定该片应放在初始 IP 数据报的哪个位置。

图 4-17 图示了一个例子。一个 4000 字节的数据报（20 字节 IP 首部加上 3980 字节 IP 有效载荷）到达一台路由器，且必须被转发到一条 MTU 为 1500 字节的链路上。这就意味着初始数据报中 3980 字节数据必须被分配为 3 个独立的片（其中的每个片也是一个 IP 数据报）。

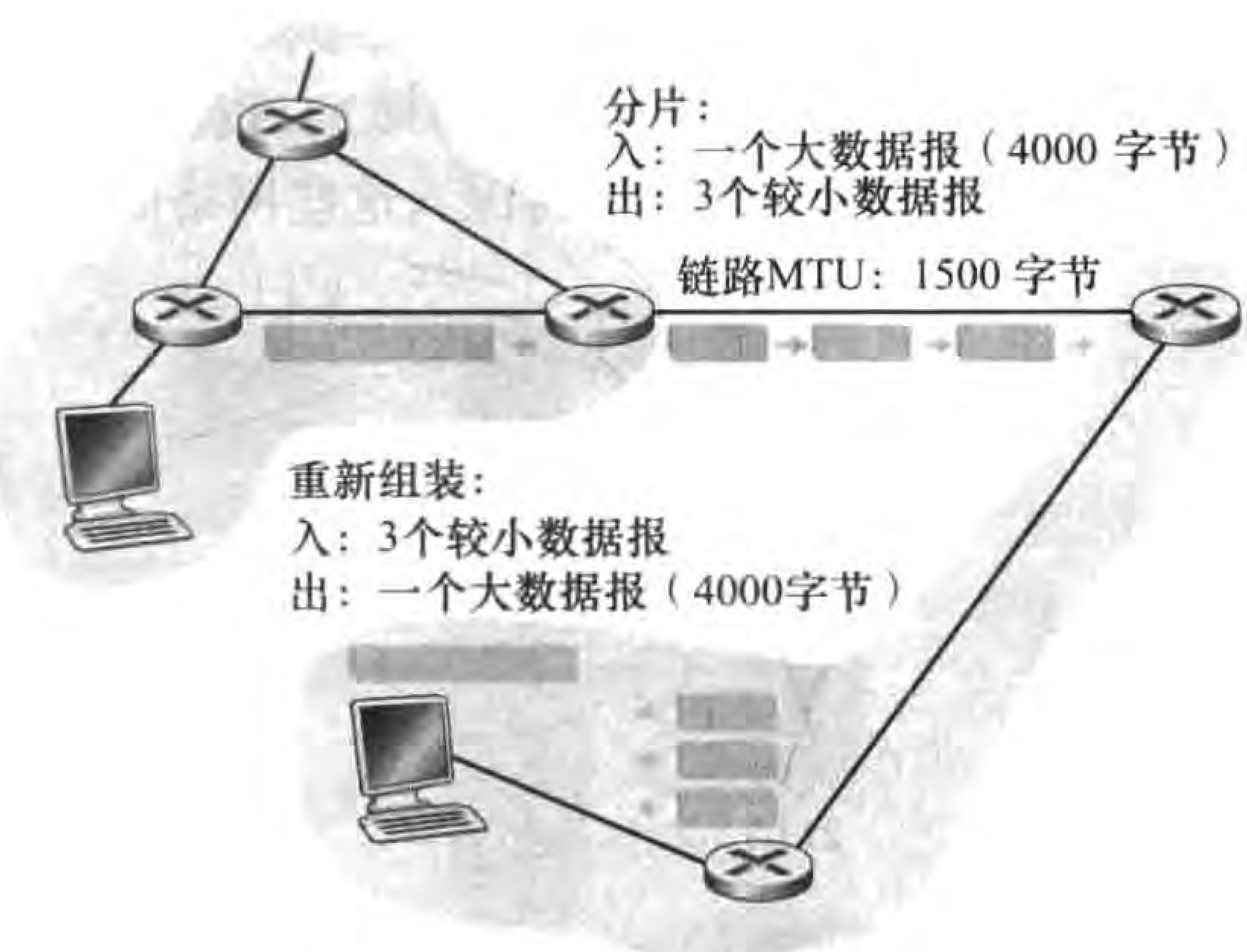


图 4-17　IP 分片与重新组装

本书的在线材料和本章后面的习题将使你能够详细探究分片。在本书的 Web 站点上，我们提供了一个 Java 小程序来产生片。提供入数据报长度、MTU 和入数据报标识，它就会自动为你产生片。参见 http://www.pearsonhighered.com/cs-resources/。

4.3.3　IPv4 编址

我们现在将注意力转向 IPv4 编址。尽管你可能认为编址是相当直接的主题，但我们希望通过本章的学习，你能认识到因特网编址不仅是一个丰富多彩、微妙和有趣的主题，而且也是一个对因特网极为重要的主题。[Stewart 1999] 的第 1 章是介绍 IPv4 编址的优秀读物。

然而，在讨论 IP 编址之前，我们需要简述一下主机与路由器连入网络的方法。一台主机通常只有一条链路连接到网络；当主机中的 IP 想发送一个数据报时，它就在该链路上发送。主机与物理链路之间的边界叫作**接口**（interface）。现在考虑一台路由器及其接口。因为路由器的任务是从链路上接收数据报并从某些其他链路转发出去，路由器必须拥有两条或更多条链路与它连接。路由器与它的任意一条链路之间的边界也叫作接口。一台路由器因此有多个接口，每个接口有其链路。因为每台主机与路由器都能发送和接收 IP 数据报，IP 要求每台主机和路由器接口拥有自己的 IP 地址。因此，从技术上讲，一个 IP 地址与一个接口相关联，而不是与包括该接口的主机或路由器相关联。

每个 IP 地址长度为 32 比特（等价为 4 字节），因此总共有 2^{32} 个（或大约 40 亿个）可能的 IP 地址。这些地址通常按所谓**点分十进制记法**（dotted-decimal notation）书写，即地址中的每个字节用它的十进制形式书写，各字节间以句点隔开。例如，考虑 IP 地址 193.32.216.9，193 是该地址的第一个 8 比特的十进制等价数，32 是该地址的第二个 8 比

特的十进制等价数，依次类推。因此，地址 193. 32. 216. 9 的二进制记法是：

11000001 00100000 11011000 00001001

在全球因特网中的每台主机和路由器上的每个接口，都必须有一个全球唯一的 IP 地址（在 NAT 后面的接口除外，在 4. 3. 4 节中讨论）。然而，这些地址不能随意地自由选择。一个接口的 IP 地址的一部分需要由其连接的子网来决定。

图 4-18 提供了一个 IP 编址与接口的例子。在该图中，一台路由器（具有 3 个接口）用于互联 7 台主机。仔细观察分配给主机和路由器接口的 IP 地址，有几点需要注意。图 4-18中左上侧的 3 台主机以及它们连接的路由器接口，都有一个形如 223. 1. 1. xxx 的 IP 地址。这就是说，在它们的 IP 地址中，最左侧的 24 比特是相同的。这 4 个接口也通过一个并不包含路由器的网络互联起来。该网络可能由一个以太网 LAN 互联，在此情况下，这些接口将通过一台以太网交换机互联（如第 6 章中讨论的那样），或者通过一个无线接入点互联（如第 7 章中讨论的那样）。我们此时将这种无路由器连接这些主机的网络表示为一朵云，在第 6、7 章中再深入这些网络的内部。

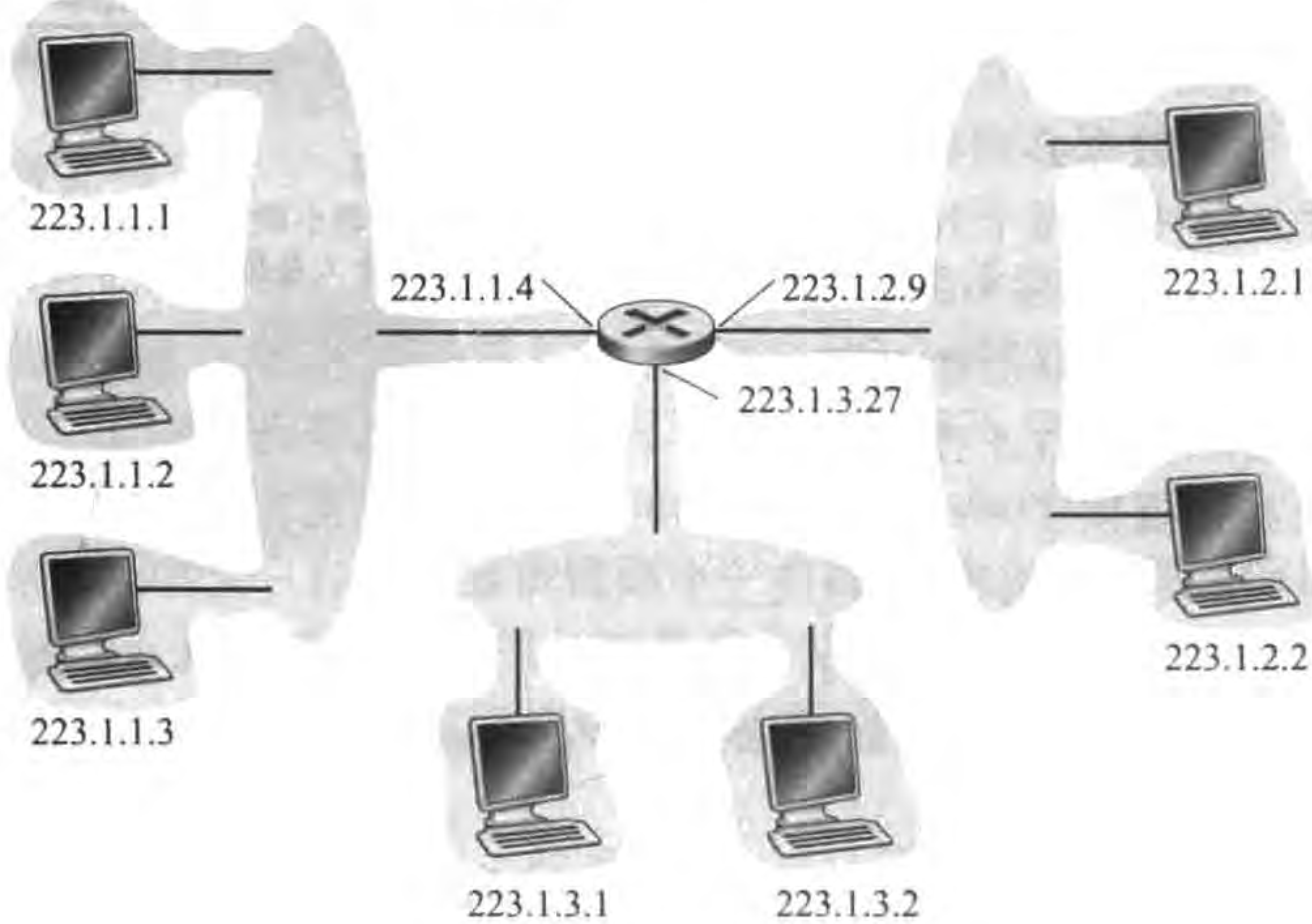


图 4-18 接口地址和子网

用 IP 的术语来说，互联这 3 个主机接口与 1 个路由器接口的网络形成一个**子网**（subnet）[RFC 950]。（在因特网文献中，子网也称为 IP 网络或直接称为网络。）IP 编址为这个子网分配一个地址 223. 1. 1. 0/24，其中的 /24 记法，有时称为**子网掩码**（network mask），指示 32 比特中的最左侧 24 比特定义了子网地址。因此子网 223. 1. 1. 0/24 由 3 个主机接口（223. 1. 1. 1、223. 1. 1. 2 和 223. 1. 1. 3）和 1 个路由器接口（223. 1. 1. 4）组成。任何其他要连到 223. 1. 1. 0/24 网络的主机都要求其地址具有 223. 1. 1. xxx 的形式。图 4-18中显示了另外两个网络：223. 1. 2. 0/24 网络与 223. 1. 3. 0/24 子网。图 4-19 图示了在

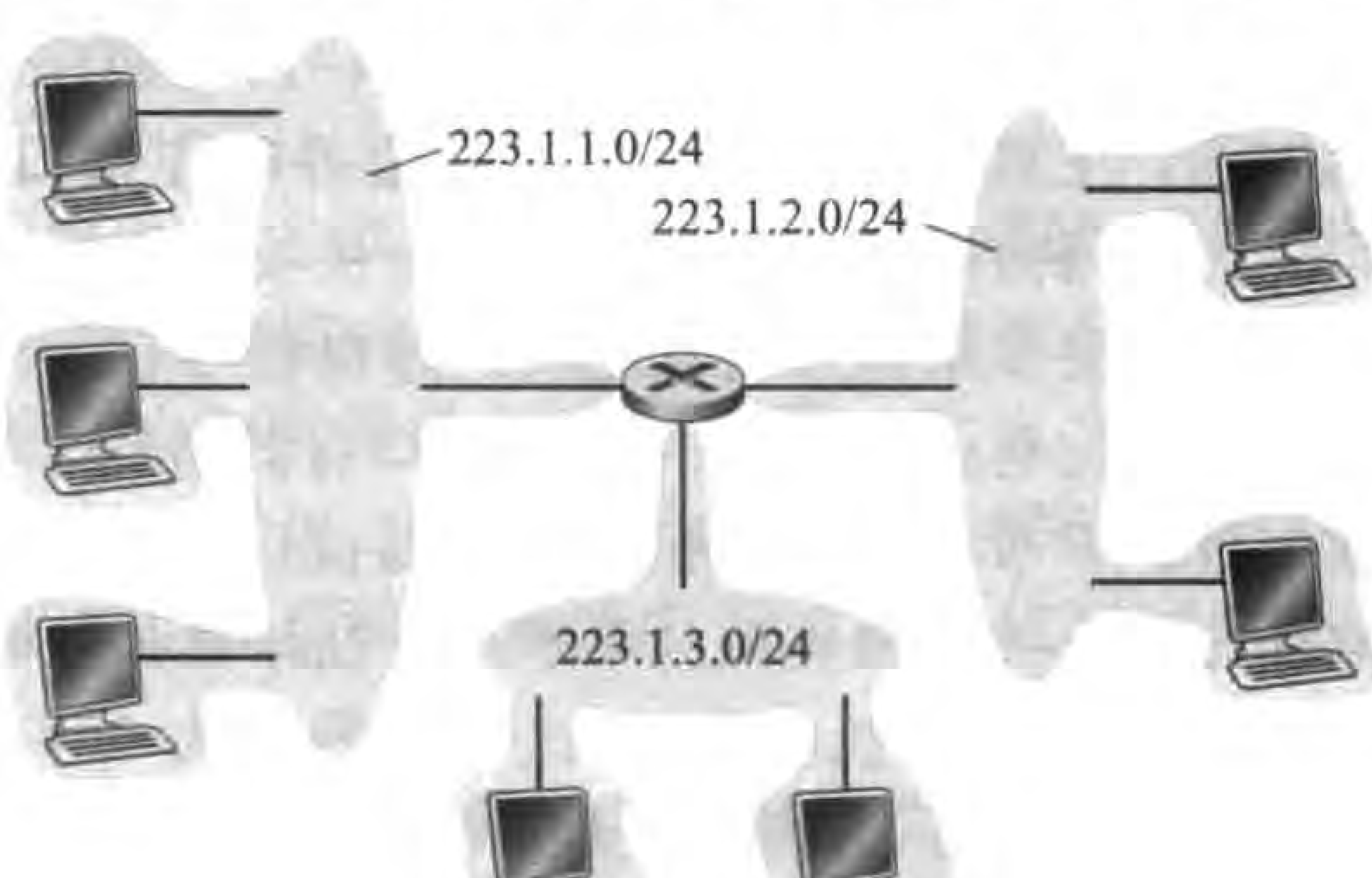


图 4-19 子网地址

图 4-18 中存在的 3 个 IP 子网。

一个子网的 IP 定义并不局限于连接多台主机到一个路由器接口的以太网段。为了搞清其中的道理，可考虑图 4-20，图中显示了 3 台通过点对点链路彼此互联的路由器。每台路由器有 3 个接口，每条点对点链路使用一个，一个用于直接将路由器连接到一对主机的广播链路。这里出现了几个子网呢？3 个子网 223.1.1.0/24、223.1.2.0/24 和 223.1.3.0/24 类似于我们在图 4-18 中遇到的子网。但注意到在本例中还有其他 3 个子网：一个子网是 223.1.9.0/24，用于连接路由器 R1 与 R2 的接口；另外一个子网是 223.1.8.0/24，用于连接路由器 R2 与 R3 的接口；第三个子网是 223.1.7.0/24，用于连接路由器 R3 与 R1 的接口。对于一个路由器和主机的通用互联系统，我们能够使用下列有效方法定义系统中的子网：

> 为了确定子网，分开主机和路由器的每个接口，产生几个隔离的网络岛，使用接口端接这些隔离的网络的端点。这些隔离的网络中的每一个都叫作一个**子网**（subnet）。

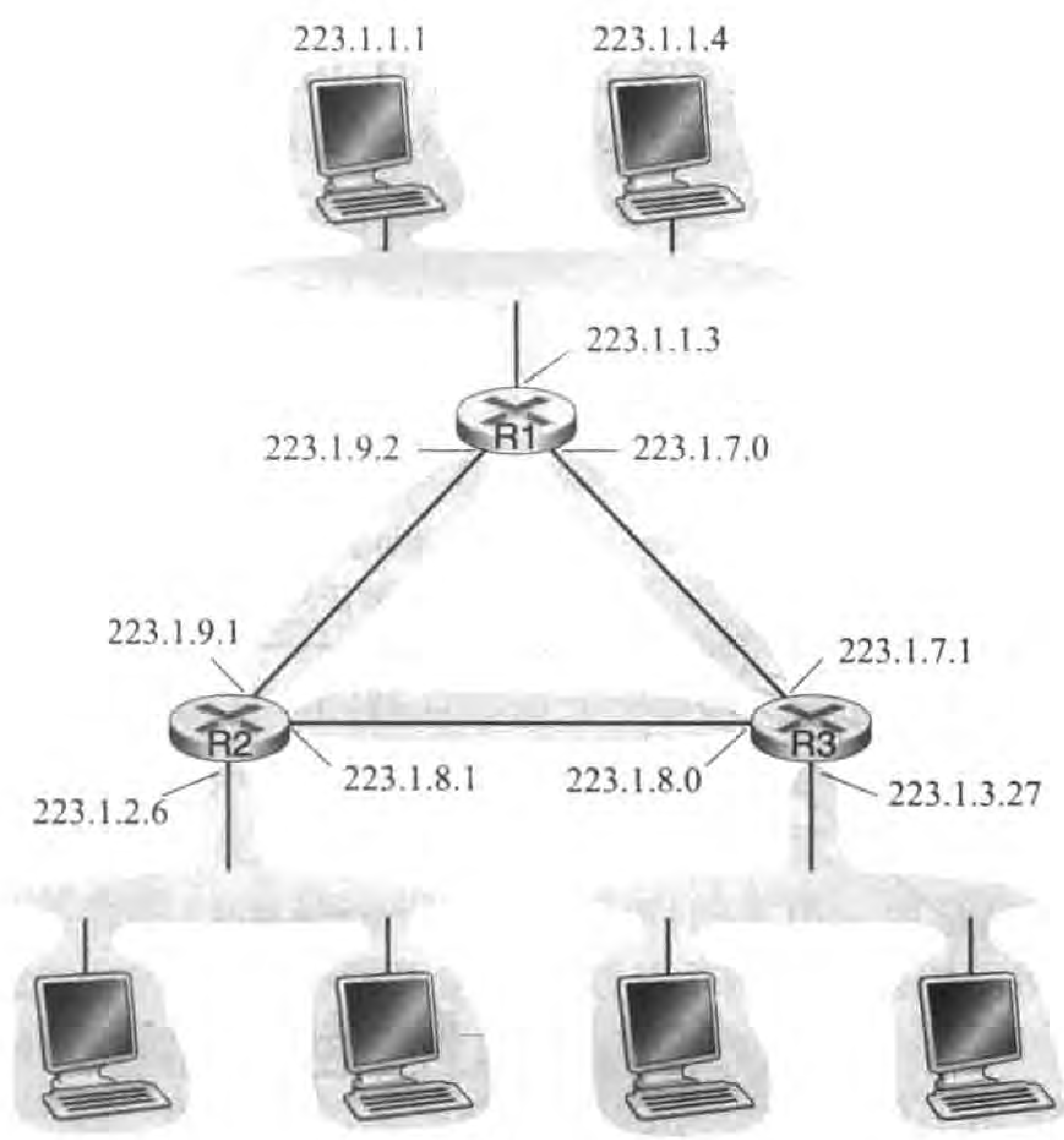


图 4-20 3 台路由器互联 6 个子网

如果我们将该过程用于图 4-20 中的互联系统上，会得到 6 个岛或子网。

从上述讨论显然可以看出，一个具有多个以太网段和点对点链路的组织（如一个公司或学术机构）将具有多个子网，在给定子网上的所有设备都具有相同的子网地址。原则上，不同的子网能够具有完全不同的子网地址。然而，在实践中，它们的子网地址经常有许多共同之处。为了理解其中的道理，我们来关注在全球因特网中是如何处理编址的。

因特网的地址分配策略被称为**无类别域间路由选择**（Classless Interdomain Routing，CIDR）[RFC 4632]。CIDR 将子网寻址的概念一般化了。当使用子网寻址时，32 比特的 IP 地址被划分为两部分，并且也具有点分十进制数形式 *a.b.c.d/x*，其中 *x* 指示了地址的第一部分中的比特数。

形式为 *a. b. c. d/x* 的地址的 *x* 最高比特构成了 IP 地址的网络部分，并且经常被称为该地址的**前缀**（prefix）（或网络前缀）。一个组织通常被分配一块连续的地址，即具有相同前缀的一段地址（参见“实践原则”）。在这种情况下，该组织内部的设备的 IP 地址将共享共同的前缀。当我们在 5.4 节中论及因特网的 BGP 路由选择协议时，将看到该组织网络外部的路由器仅考虑前面的前缀比特 *x*。这就是说，当该组织外部的一台路由器转发一个数据报，且该数据报的目的地址位于该组织的内部时，仅需要考虑该地址的前面 *x* 比特。这相当大地减少了在这些路由器中转发表的长度，因为形式为 *a. b. c. d/x* 的单一表项足以将数据报转发到该组织内的任何目的地。

实践原则

这是一个 ISP 将 8 个组织连接到因特网的例子，它也很好地说明了仔细分配 CIDR 化的地址有利于路由选择的道理。如图 4-21 所示，假设该 ISP（我们称之为 Fly-By-Night-ISP）向外界通告，它应该发送所有地址的前 20 比特与 200. 23. 16. 0/20 相符的数据报。外界的其他部分不需要知道在地址块 200. 23. 16. 0/20 内实际上还存在 8 个其他组织，其中每个组织有自己的子网。这种使用单个网络前缀通告多个网络的能力通常称为**地址聚合**（address aggregation），也称为**路由聚合**（route aggregation）或**路由摘要**（route summarization）。

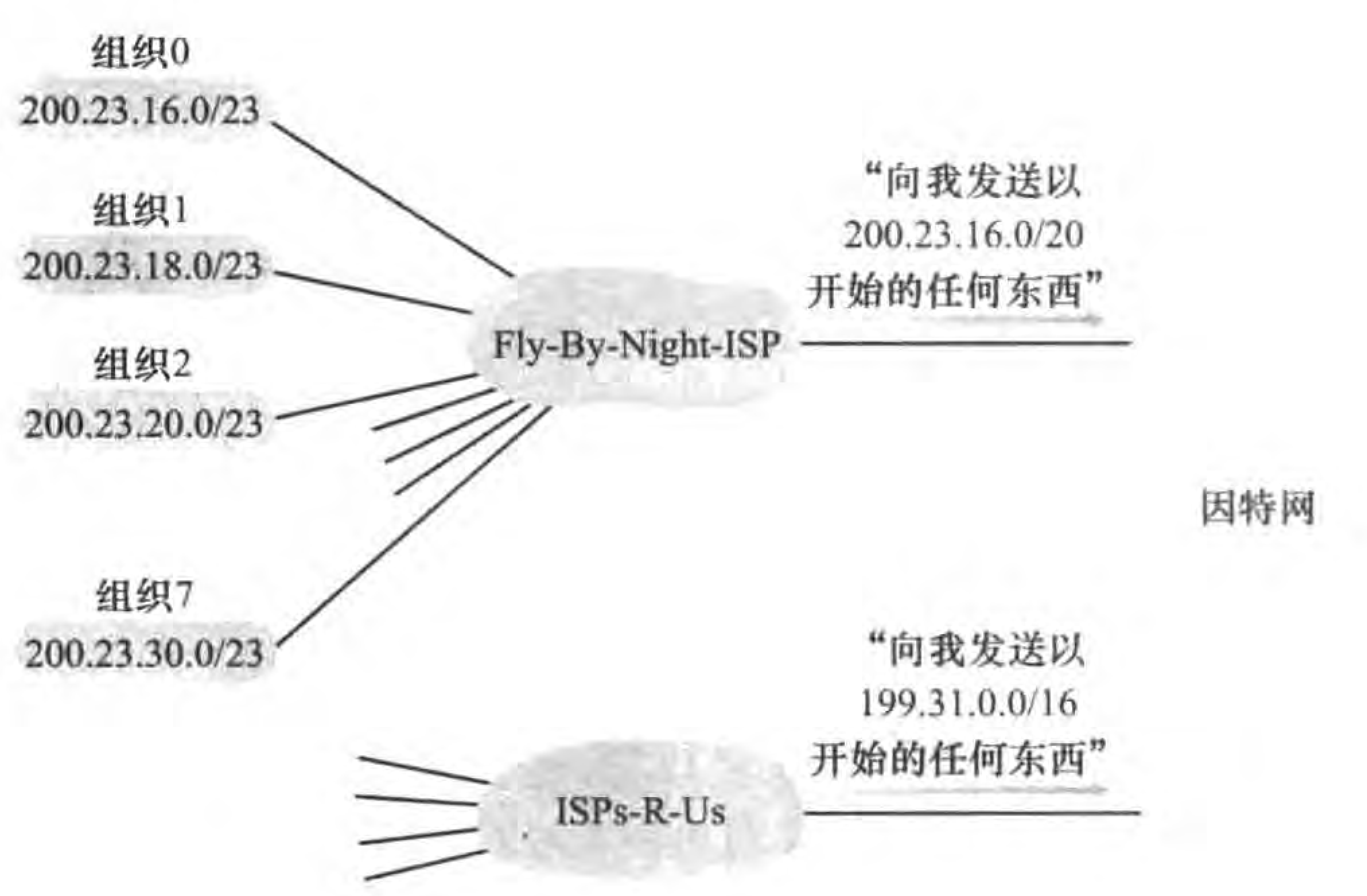


图 4-21　层次编址与路由聚合

当地址按块分给 ISP，然后又由 ISP 分给客户组织时，地址聚合工作极为有效。但是当地址不是按这样的层次方式分配时，会出现什么情况呢？例如，如果 Fly-By-Night-ISP 获取了 ISPs-R-Us，然后让组织 1 通过它辅助的 ISPs-R-Us 与因特网相连，将会发生什么情况呢？如图 4-21 中所示，该辅助的 ISPs-R-Us 拥有地址块 199. 31. 0. 0/16，但很遗憾的是组织 1 的 IP 地址在该地址块之外。这里可以采取什么措施呢？组织 1 无疑可以将其所有的路由器和主机重新编号，使得地址在 ISPs-R-Us 的地址块内。但这是一种代价很高的方案，而且组织 1 将来也许还会从 ISPs-R-Us 更换到另一个 ISP。采用的典型方案是，组织 1 保持其 IP 地址在 200. 23. 18. 0/23 内。在这种情况下，如图 4-22 所示，

Fly-By-Night-ISP 继续通告地址块 200.23.16.0/20，并且 ISPs-R-Us 也继续通告地址块 199.31.0.0/16。然而，ISPs-R-Us 现在还要通告组织 1 的地址块 200.23.18.0/23。当在更大的因特网上的其他路由器看见地址块 200.23.16.0/20（来自 Fly-By-Night-ISP）和 200.23.18.0/23（来自 ISPs-R-Us），并且想路由选择到地址块 200.23.18.0/23 内的一个地址时，它们将使用**最长前缀匹配**（参见 4.2.1 节），并朝着 ISPs-R-Us 路由，因为它通告了与目的地址相匹配的最长（最具体）的地址前缀。

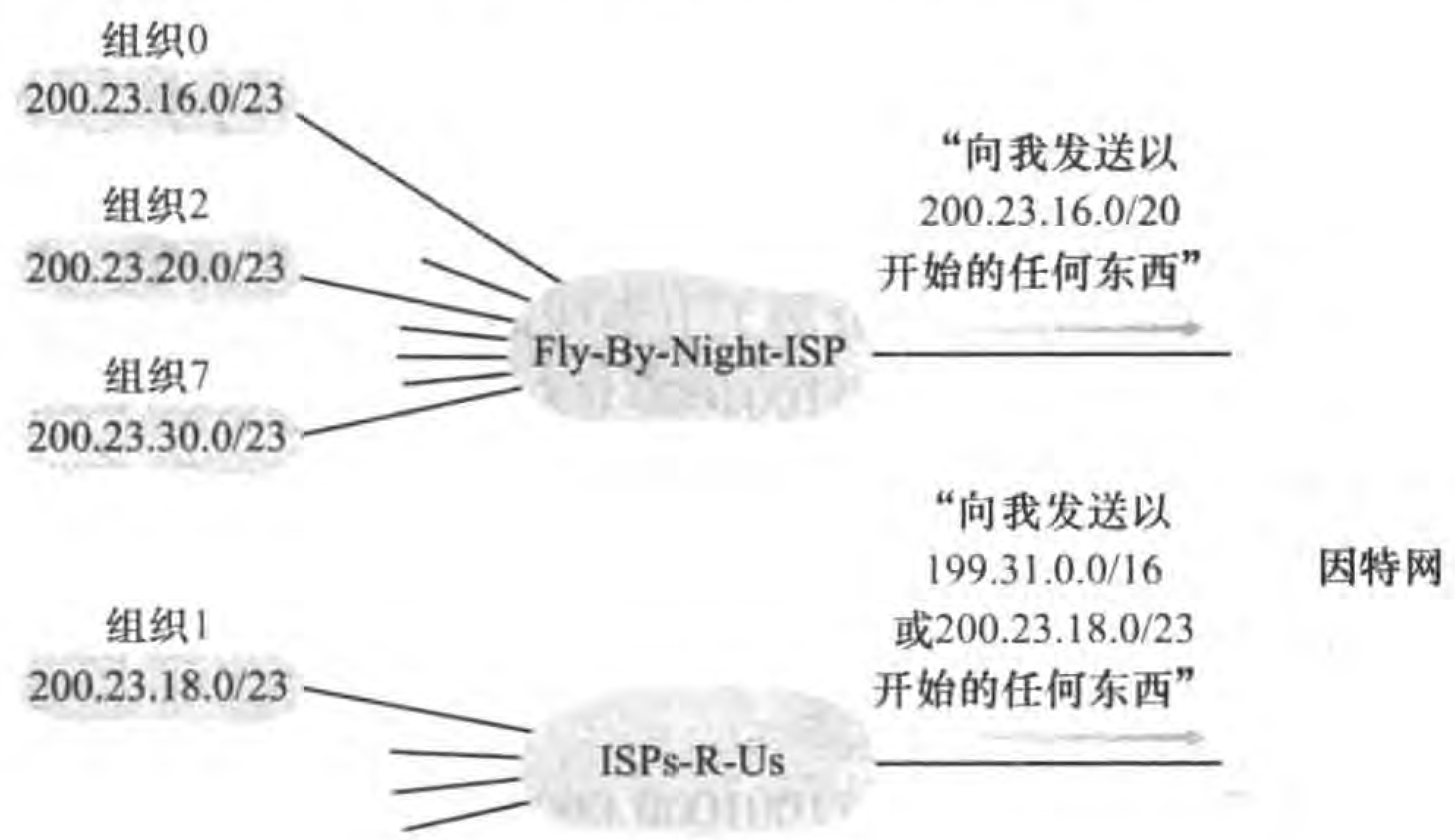


图 4-22 ISPs-R-Us 具有到组织 1 的一条更具体的路由

一个地址的剩余 32 - x 比特可认为是用于区分该组织内部设备的，其中的所有设备具有相同的网络前缀。当该组织内部的路由器转发分组时，才会考虑这些比特。这些较低阶比特可能（或可能不）具有另外的子网结构，如前面所讨论的那样。例如，假设某 CIDR 化的地址 *a.b.c.d*/21 的前 21 比特定义了该组织的网络前缀，它对该组织中所有主机的 IP 地址来说是共同的。其余的 11 比特标识了该组织内的主机。该组织的内部结构可以采用这样的方式，使用最右边的 11 比特在该组织中划分子网，就像前面所讨论的那样。例如，*a.b.c.d*/24 可能表示该组织内的特定子网。

在 CIDR 被采用之前，IP 地址的网络部分被限制为长度为 8、16 或 24 比特，这是一种称为**分类编址**（classful addressing）的编址方案，这是因为具有 8、16 和 24 比特子网地址的子网分别被称为 A、B 和 C 类网络。一个 IP 地址的网络部分正好为 1、2 或 3 字节的要求，已经在支持数量迅速增加的具有小规模或中等规模子网的组织方面出现了问题。一个 C 类（/24）子网仅能容纳多达 $2^8-2=254$（$2^8=256$，其中的两个地址预留用于特殊用途）台主机，这对于许多组织来说太小了。然而一个 B 类（/16）子网可支持多达 65 534 台主机，又太大了。在分类编址方法下，比方说一个有 2000 台主机的组织通常被分给一个 B 类（/16）地址。这就导致了 B 类地址空间的迅速损耗以及所分配的地址空间的利用率低下。例如，为具有 2000 台主机的组织分配一个 B 类地址，就具有足以支持多达 65 534 个接口的地址空间，剩下的超过 63 000 个地址却不能被其他组织使用。

如果还不提及另一种类型的 IP 地址，即 IP 广播地址 255.255.255.255，那将是我们的疏漏。当一台主机发出一个目的地址为 255.255.255.255 的数据报时，该报文会交付给

同一个网络中的所有主机。路由器也会有选择地向邻近的子网转发该报文（虽然它们通常不这样做）。

现在我们已经详细地学习了 IP 编址，需要知道主机或子网最初是如何得到它们的地址的。我们先看一个组织是如何为其设备得到一个地址块的，然后再看一个设备（如一台主机）是如何从某组织的地址块中分配到一个地址的。

1. 获取一块地址

为了获取一块 IP 地址用于一个组织的子网内，某网络管理员也许首先会与他的 ISP 联系，该 ISP 可能会从已分给它的更大地址块中提供一些地址。例如，该 ISP 也许自己已被分配了地址块 200.23.16.0/20。该 ISP 可以依次将该地址块分成 8 个长度相等的连续地址块，为本 ISP 支持的最多达 8 个组织中的一个分配这些地址块中的一块，如下所示。（为了便于查看，我们已将这些地址的网络部分加了下划线。）

ISP 的地址块	200.23.16.0/20	11001000 00010111 00010000 00000000
组织 0	200.23.16.0/23	11001000 00010111 00010000 00000000
组织 1	200.23.18.0/23	11001000 00010111 00010010 00000000
组织 2	200.23.20.0/23	11001000 00010111 00010100 00000000
……	……	……
组织 7	200.23.30.0/23	11001000 00010111 00011110 00000000

尽管从一个 ISP 获取一组地址是一种得到一块地址的方法，但这不是唯一的方法。显然，必须还有一种方法供 ISP 本身得到一块地址。是否有一个全球性的权威机构，它具有管理 IP 地址空间并向各 ISP 和其他组织分配地址块的最终责任呢？的确有一个！IP 地址由因特网名字和编号分配机构（Internet Corporation for Assigned Names and Numbers，ICANN）[ICANN 2016] 管理，管理规则基于 [RFC 7020]。非营利的 ICANN 组织 [NTIA 1998] 的作用不仅是分配 IP 地址，还管理 DNS 根服务器。它还有一项容易引起争论的工作，即分配域名与解决域名纷争。ICANN 向区域性因特网注册机构（如 ARIN、RIPE、APNIC 和 LACNIC）分配地址，这些机构一起形成了 ICANN 的地址支持组织 [ASO-ICANN 2016]，处理本区域内的地址分配/管理。

2. 获取主机地址：动态主机配置协议

某组织一旦获得了一块地址，它就可为本组织内的主机与路由器接口逐个分配 IP 地址。系统管理员通常手工配置路由器中的 IP 地址（常常在远程通过网络管理工具进行配置）。主机地址也能手动配置，但是这项任务目前更多的是使用**动态主机配置协议**（Dynamic Host Configuration，DHCP）[RFC 2131] 来完成。DHCP 允许主机自动获取（被分配）一个 IP 地址。网络管理员能够配置 DHCP，以使某给定主机每次与网络连接时能得到一个相同的 IP 地址，或者某主机将被分配一个**临时的 IP 地址**（temporary IP address），每次与网络连接时该地址也许是不同的。除了主机 IP 地址分配外，DHCP 还允许一台主机得知其他信息，例如它的子网掩码、它的第一跳路由器地址（常称为默认网关）与它的本地 DNS 服务器的地址。

由于 DHCP 具有将主机连接进一个网络的网络相关方面的自动能力，故它又常被称为**即插即用协议**（plug-and-play protocol）或**零配置**（zeroconf）**协议**。这种能力对于网络管理员来说非常有吸引力，否则他将不得不手工执行这些任务！DHCP 还广泛地用于住宅因特网接入网、企业网与无线局域网中，其中的主机频繁地加入和离开网络。例如，考虑一个学

生带着便携机从宿舍到图书馆再到教室。很有可能在每个位置这个学生将连接到一个新的子网，因此在每个位置都需要一个新的 IP 地址。DHCP 是适合这种情形的理想方法，因为有许多用户来来往往，并且仅在有限的时间内需要地址。DHCP 的即插即用能力的价值是显然的，因为下列情况是不可想象的：系统管理员在每个位置能够重新配置便携机，并且少数学生（除了那些上过计算机网络课程的学生）让专家人工地配置他们的便携机。

DHCP 是一个客户－服务器协议。客户通常是新到达的主机，它要获得包括自身使用的 IP 地址在内的网络配置信息。在最简单场合下，每个子网（在图 4-20 的编址意义下）将具有一台 DHCP 服务器。如果在某子网中没有服务器，则需要一个 DHCP 中继代理（通常是一台路由器），这个代理知道用于该网络的 DHCP 服务器的地址。图 4-23 显示了连接到子网 223.1.2/24 的一台 DHCP 服务器，具有一台提供中继代理服务的路由器，它为连接到子网 223.1.1/24 和 223.1.3/24 的到达客户提供 DHCP 服务。在我们下面的讨论中，将假定 DHCP 服务器在该子网上是可供使用的。

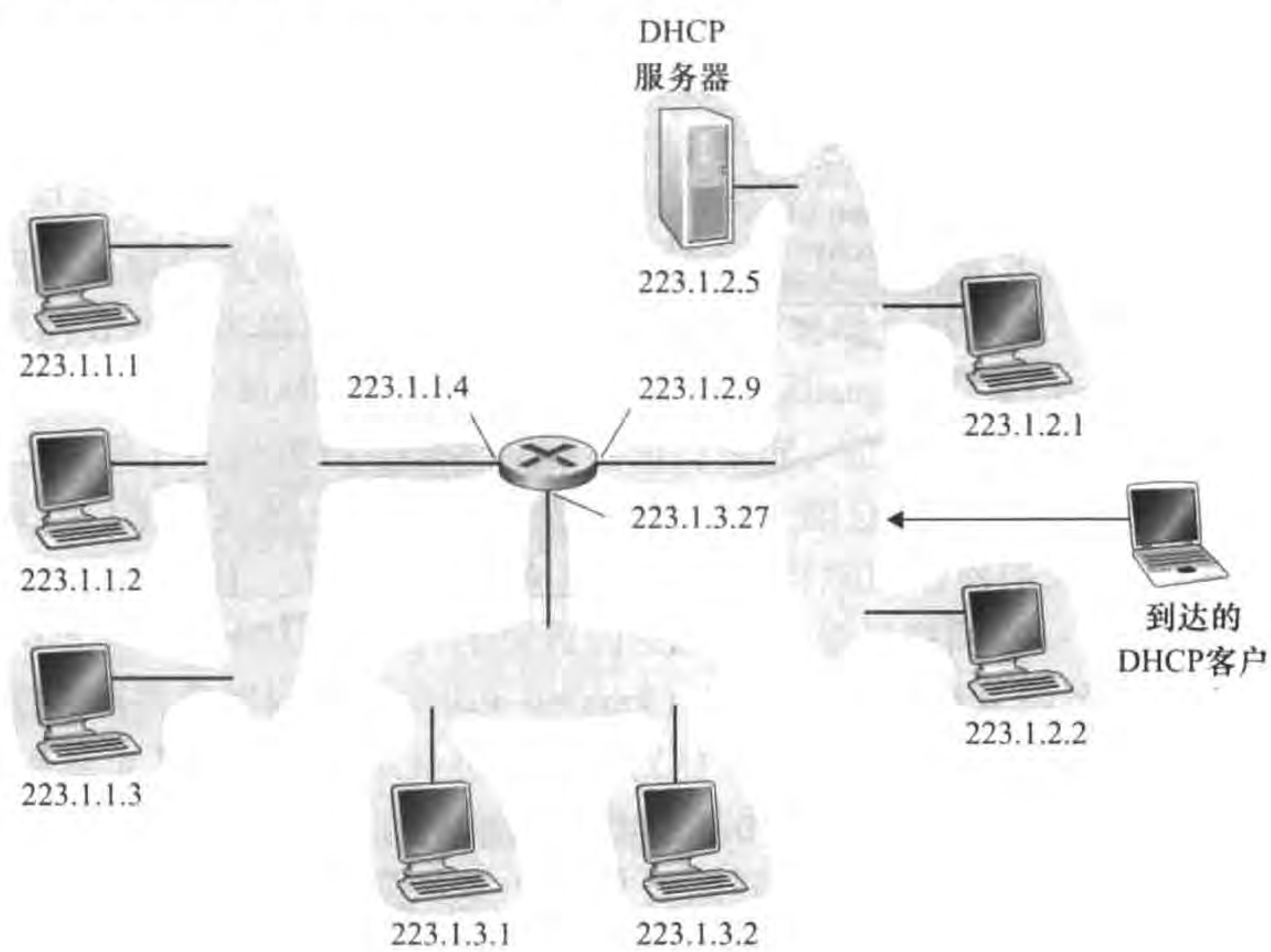


图 4-23 DHCP 客户和服务器

对于一台新到达的主机而言，针对图 4-23 所示的网络设置，DHCP 协议是一个 4 个步骤的过程，如图 4-24 中所示。在这幅图中，yiaddr（表示“你的因特网地址”之意）指示分配给该新到达客户的地址。

这 4 个步骤是：

- DHCP 服务器发现。一台新到达的主机的首要任务是发现一个要与其交互的 DHCP 服务器。这可通过使用 **DHCP 发现报文**（DHCP discover message）来完成，客户在 UDP 分组中向端口 67 发送该发现报文。该 UDP 分组封装在一个 IP 数据报中。但是这个数据报应发给谁呢？主机甚至不知道它所连接网络的 IP 地址，更不用说用于该网络的 DHCP 服务器地址了。在这种情况下，DHCP 客户生成包含 DHCP 发现报文的 IP 数据报，其中使用广播目的地址 255.255.255.255 并且使用“本主机”源 IP 地址 0.0.0.0。DHCP 客户将该 IP 数据报传递给链路层，链路层然后将该帧广播到所有与

该子网连接的节点（我们将在6.4节中涉及链路层广播的细节）。

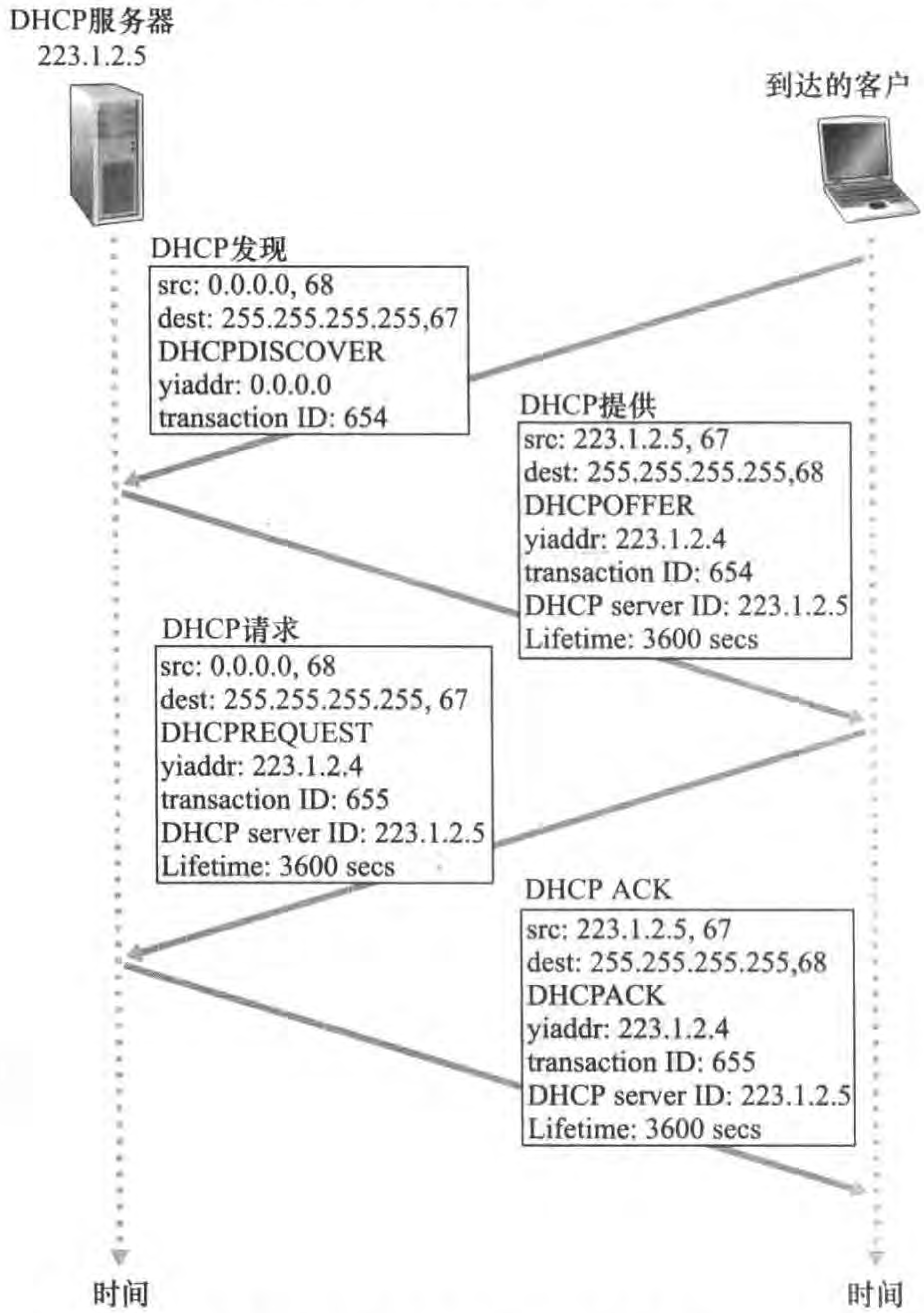


图4-24　DHCP客户-服务器交互

- DHCP服务器提供。DHCP服务器收到一个DHCP发现报文时，用**DHCP提供报文**（DHCP offer message）向客户做出响应，该报文向该子网的所有节点广播，仍然使用IP广播地址255.255.255.255（你也许要思考一下这个服务器为何也必须采用广播）。因为在子网中可能存在几个DHCP服务器，该客户也许会发现它处于能在几个提供者之间进行选择的优越位置。每台服务器提供的报文包含有收到的发现报文的事务ID、向客户推荐的IP地址、网络掩码以及**IP地址租用期**（address lease time），即IP地址有效的时间量。服务器租用期通常设置为几小时或几天［Droms 2002］。
- DHCP请求。新到达的客户从一个或多个服务器提供中选择一个，并向选中的服务器提供用**DHCP请求报文**（DHCP request message）进行响应，回显配置的参数。
- DHCP ACK。服务器用**DHCP ACK报文**（DHCP ACK message）对DHCP请求报文进行响应，证实所要求的参数。

一旦客户收到DHCP ACK后，交互便完成了，并且该客户能够在租用期内使用DHCP

分配的 IP 地址。因为客户可能在该租用期超时后还希望使用这个地址，所以 DHCP 还提供了一种机制以允许客户更新它对一个 IP 地址的租用。

从移动性角度看，DHCP 确实有非常严重的缺陷。因为每当节点连到一个新子网，要从 DHCP 得到一个新的 IP 地址，当一个移动节点在子网之间移动时，就不能维持与远程应用之间的 TCP 连接。在第 6 章中，我们将研究移动 IP，它是一种对 IP 基础设施的扩展，允许移动节点在网络之间移动时使用其单一永久的地址。有关 DHCP 的其他细节可在 [Droms 2002] 与 [dhc 2016] 中找到。一个 DHCP 的开放源码参考实现可从因特网系统协会 [ISC 2016] 得到。

4.3.4 网络地址转换

讨论了有关因特网地址和 IPv4 数据报格式后，我们现在可清楚地认识到每个 IP 使能的设备都需要一个 IP 地址。随着所谓小型办公室、家庭办公室（Small Office，Home Office，SOHO）子网的大量出现，看起来意味着每当一个 SOHO 想安装一个 LAN 以互联多台机器时，需要 ISP 分配一组地址以供该 SOHO 的所有 IP 设备（包括电话、平板电脑、游戏设备、IP TV、打印机等）使用。如果该子网变大了，则需要分配一块较大的地址。但如果 ISP 已经为 SOHO 网络的当前地址范围分配过一块连续地址该怎么办呢？并且，家庭主人一般要（或应该需要）首先知道的管理 IP 地址的典型方法有哪些呢？幸运的是，有一种简单的方法越来越广泛地用在这些场合：**网络地址转换**（Network Address Translation，NAT）[RFC 2663；RFC 3022；Huston 2004；Zhang 2007；Cisco NAT 2016]。

图 4-25 显示了一台 NAT 使能路由器的运行情况。位于家中的 NAT 使能的路由器有一个接口，该接口是图 4-25 中右侧所示家庭网络的一部分。在家庭网络内的编址就像我们在上面看到的完全一样，其中的所有 4 个接口都具有相同的网络地址 10.0.0/24。地址空间 10.0.0.0/8 是在 [RFC 1918] 中保留的三部分 IP 地址空间之一，这些地址用于如图 4-25 中的家庭网络等**专用网络**（private network）或**具有专用地址的地域**（realm with private address）。具有专用地址的地域是指其地址仅对该网络中的设备有意义的网络。为了明白它为什么重要，考虑有数十万家庭网络这样的事实，许多使用了相同的地址空间 10.0.0.0/24。在一个给定家庭网络中的设备能够使用 10.0.0.0/24 编址彼此发送分组。然而，转发到家庭网络之外进入更大的全球因特网的分组显然不能使用这些地址（或作为源地址，或作为目的地址），因为有数十万的网络使用着这块地址。这就是说，10.0.0.0/24 地址仅在给定的网络中才有意义。但是如果专用地址仅在给定的网络中才有意义的话，当向或从全球因特网发送或接收分组时如何处理编址问题呢，地址在何处才必须是唯一的呢？答案在于理解 NAT。

NAT 使能路由器对于外部世界来说甚至不像一台路由器。相反 NAT 路由器对外界的行为就如同一个具有单一 IP 地址的单一设备。在图 4-25 中，所有离开家庭路由器流向更大因特网的报文都拥有一个源 IP 地址 138.76.29.7，且所有进入家庭的报文都拥有同一个目的 IP 地址 138.76.29.7。从本质上讲，NAT 使能路由器对外界隐藏了家庭网络的细节。（另外，你也许想知道家庭网络计算机是从哪儿得到其地址，路由器又是从哪儿得到它的单一 IP 地址的。在通常的情况下，答案是相同的，即 DHCP！路由器从 ISP 的 DHCP 服务器得到它的地址，并且路由器运行一个 DHCP 服务器，为位于 NAT-DHCP 路由器控制的家庭网络地址空间中的计算机提供地址。）

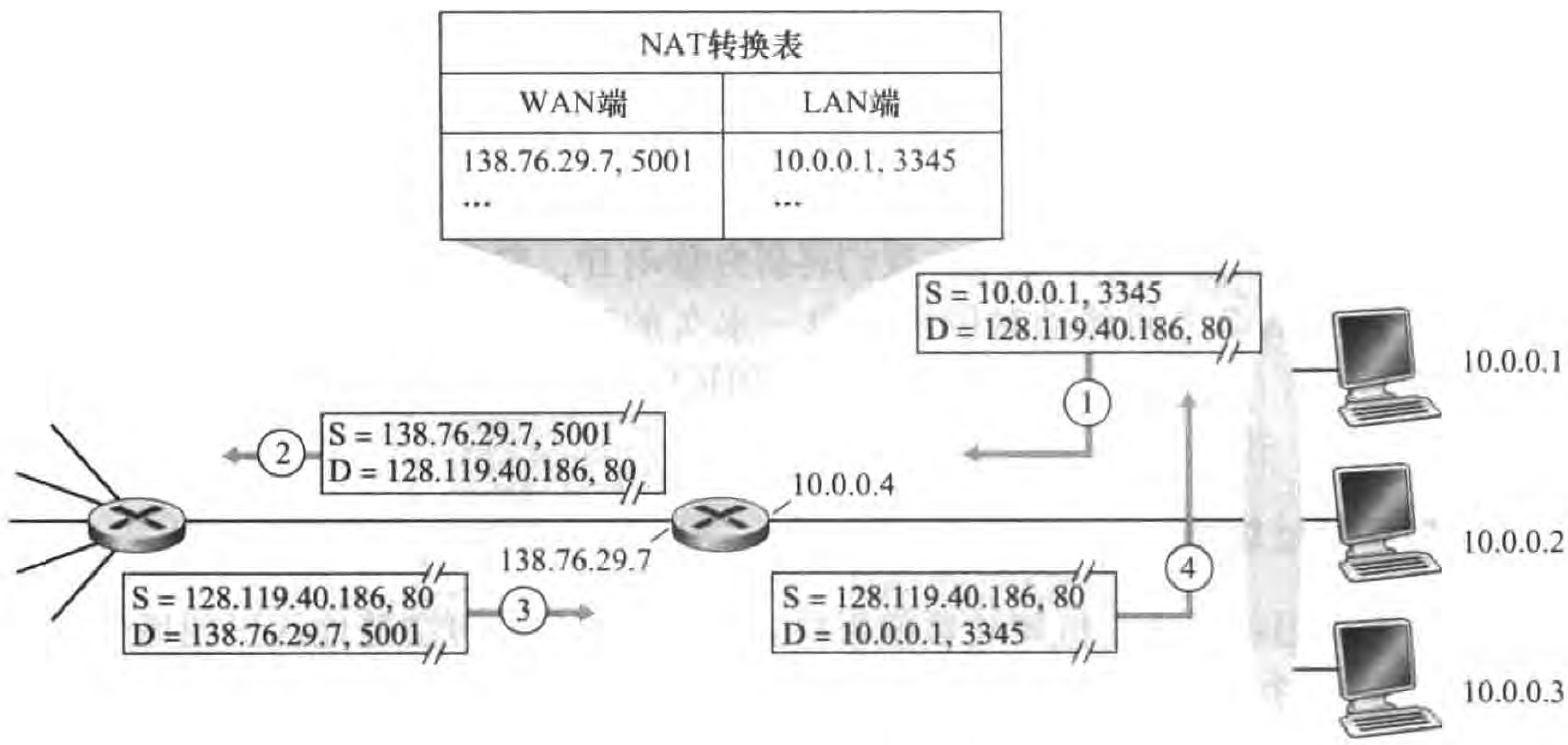


图4-25 网络地址转换

如果从广域网到达NAT路由器的所有数据报都有相同的目的IP地址（特别是对NAT路由器广域网一侧的接口），那么该路由器怎样知道它应将某个分组转发给哪个内部主机呢？技巧就是使用NAT路由器上的一张**NAT转换表**（NAT translation table），并且在表项中包含了端口号及其IP地址。

考虑图4-25中的例子。假设一个用户坐在家庭网络主机10.0.0.1后，请求IP地址为128.119.40.186的某台Web服务器（端口80）上的一个Web页面。主机10.0.0.1为其指派了（任意）源端口号3345并将该数据报发送到LAN中。NAT路由器收到该数据报，为该数据报生成一个新的源端口号5001，将源IP替代为其广域网一侧接口的IP地址138.76.29.7，且将源端口3345更换为新端口5001。当生成一个新的源端口号时，NAT路由器可选择任意一个当前未在NAT转换表中的源端口号。（注意到因为端口号字段为16比特长，NAT协议可支持超过60 000个并行使用路由器广域网一侧单个IP地址的连接！）路由器中的NAT也在它的NAT转换表中增加一表项。Web服务器并不知道刚到达的包含HTTP请求的数据报已被NAT路由器进行了改装，它会发回一个响应报文，其目的地址是NAT路由器的IP地址，其目的端口是5001。当该报文到达NAT路由器时，路由器使用目的IP地址与目的端口号从NAT转换表中检索出家庭网络浏览器使用的适当IP地址（10.0.0.1）和目的端口号（3345）。于是，路由器重写该数据报的目的IP地址与目的端口号，并向家庭网络转发该数据报。

NAT在近年来已得到了广泛的应用。但是NAT并非没有贬低者。首先，有人认为端口号是用于进程寻址的，而不是用于主机寻址的。这种违规用法对于运行在家庭网络中的服务器来说确实会引起问题，因为正如我们在第2章所见，服务器进程在周知端口号上等待入请求，并且P2P协议中的对等方在充当服务器时需要接受入连接。对这些问题的技术解决方案包括**NAT穿越**（NAT traversal）工具［RFC 5389］和通用即插即用（Universal Plug and Play，UPnP）。UPnP是一种允许主机发现和配置邻近NAT的协议［UPnP Forum 2016］。

其次，纯粹的体系结构者提出了更为“原理性的”反对NAT的意见。这时，关注焦点在于路由器是指第三层（即网络层）设备，并且应当处理只能达到网络层的分组。NAT

违反主机应当直接彼此对话这个原则，没有干涉节点修改 IP 地址，更不用说端口号。但不管喜欢与否，NAT 已成为因特网的一个重要组件，成为所谓**中间盒**［Sekar 2011］，它运行在网络层并具有与路由器十分不同的功能。中间盒并不执行传统的数据报转发，而是执行诸如 NAT、流量流的负载均衡、流量防火墙（参见下面的插入框内容）等功能。我们将在随后的 4.4 节学习的通用转发范例，除了传统的路由器转发外，还允许一些这样的中间盒功能，从而以通用、综合的方式完成转发。

关注安全性

检查数据报：防火墙和入侵检测系统

假定你被赋予了管理家庭网络、部门网络、大学网络或公司网络的任务。知道你网络 IP 地址范围的攻击者，能够方便地在此范围中发送 IP 数据报进行寻址。这些数据报能够做各种不正当的事情，包括用 ping 搜索和端口扫描形成你的网络图，用恶意分组使易受攻击的主机崩溃，扫描你网络中服务器上的开放 TCP/UDP 端口，并且通过在分组中带有恶意软件来感染主机。作为网络管理员，你准备做些什么来将这些能够在你的网络中发送恶意分组的坏家伙拒之门外呢？对抗恶意分组攻击的两种流行的防御措施是防火墙和入侵检测系统（IDS）。

作为一名网络管理员，你可能首先尝试在你的网络和因特网之间安装一台防火墙。（今天大多数接入路由器具有防火墙能力。）防火墙检查数据报和报文段首部字段，拒绝可疑的数据报进入内部网络。例如，一台防火墙可以被配置为阻挡所有的 ICMP 回显请求分组（参见 5.6 节），从而防止了攻击者横跨你的 IP 地址范围进行传统的端口扫描。防火墙也能基于源和目的 IP 地址以及端口号阻挡分组。此外，防火墙能够配置为跟踪 TCP 连接，仅许可属于批准连接的数据报进入。

IDS 能够提供另一种保护措施。IDS 通常位于网络的边界，执行“深度分组检查”，不仅检查数据报（包括应用层数据）中的首部字段，而且检查其有效载荷。IDS 具有一个分组特征数据库，这些特征是已知攻击的一部分。随着新攻击的发现，该数据库自动更新特征。当分组通过 IDS 时，IDS 试图将分组的首部字段和有效载荷与其特征数据库中的特征相匹配。如果发现了这样的一种匹配，就产生一个告警。入侵防止系统（IPS）与 IDS 类似，只是除了产生告警外还实际阻挡分组。在第 8 章中，我们将更为详细地研究防火墙和 IDS。

防火墙和 IDS 能够全面保护你的网络免受所有攻击吗？答案显然是否定的，因为攻击者继续寻找特征还不能匹配的新攻击方法。但是防火墙和传统的基于特征的 IDS 在保护你的网络不受已知攻击入侵方面是有用的。

4.3.5 IPv6

在 20 世纪 90 年代早期，因特网工程任务组就开始致力于开发一种替代 IPv4 的协议。该努力的首要动机是以下现实：由于新的子网和 IP 节点以惊人的增长率连到因特网上（并被分配唯一的 IP 地址），32 比特的 IP 地址空间即将用尽。为了应对这种对大 IP 地址空间的需求，开发了一种新的 IP 协议，即 IPv6。IPv6 的设计者还利用这次机会，在 IPv4 积累的运行经验基础上加进和强化了 IPv4 的其他方面。

IPv4 地址在什么时候会被完全分配完（因此没有新的网络再能与因特网相连）是一个相当有争议的问题。IETF 的地址寿命期望工作组的两位负责人分别估计地址将于 2008 年和 2018 年用完［Solensky 1996］。在 2011 年 2 月，IANA 向一个区域注册机构分配完了未分配 IPv4 地址的最后剩余地址池。这些注册机构在它们的地址池中还有可用的 IPv4 地址，一旦用完这些地址，从中央池中将再也分配不出更多的可用地址块了［Huston 2011a］。IPv4 地址空间耗尽的近期调研以及延长该地址空间的寿命所采取的步骤见［Richter 2015］。

尽管在 20 世纪 90 年代中期对 IPv4 地址耗尽的估计表明，IPv4 地址空间耗尽的期限还有可观的时间，但人们认识到，如此大规模地部署一项新技术将需要可观的时间，因此研发 IP 版本 6（IPv6）［RFC 2460］的工作开始了［RFC 1752］。（一个经常问的问题是：IPv5 出了什么情况？人们最初预想 ST-2 协议将成为 IPv5，但 ST-2 后来被舍弃了。）有关 IPv6 的优秀信息来源见［Huitema 1998］。

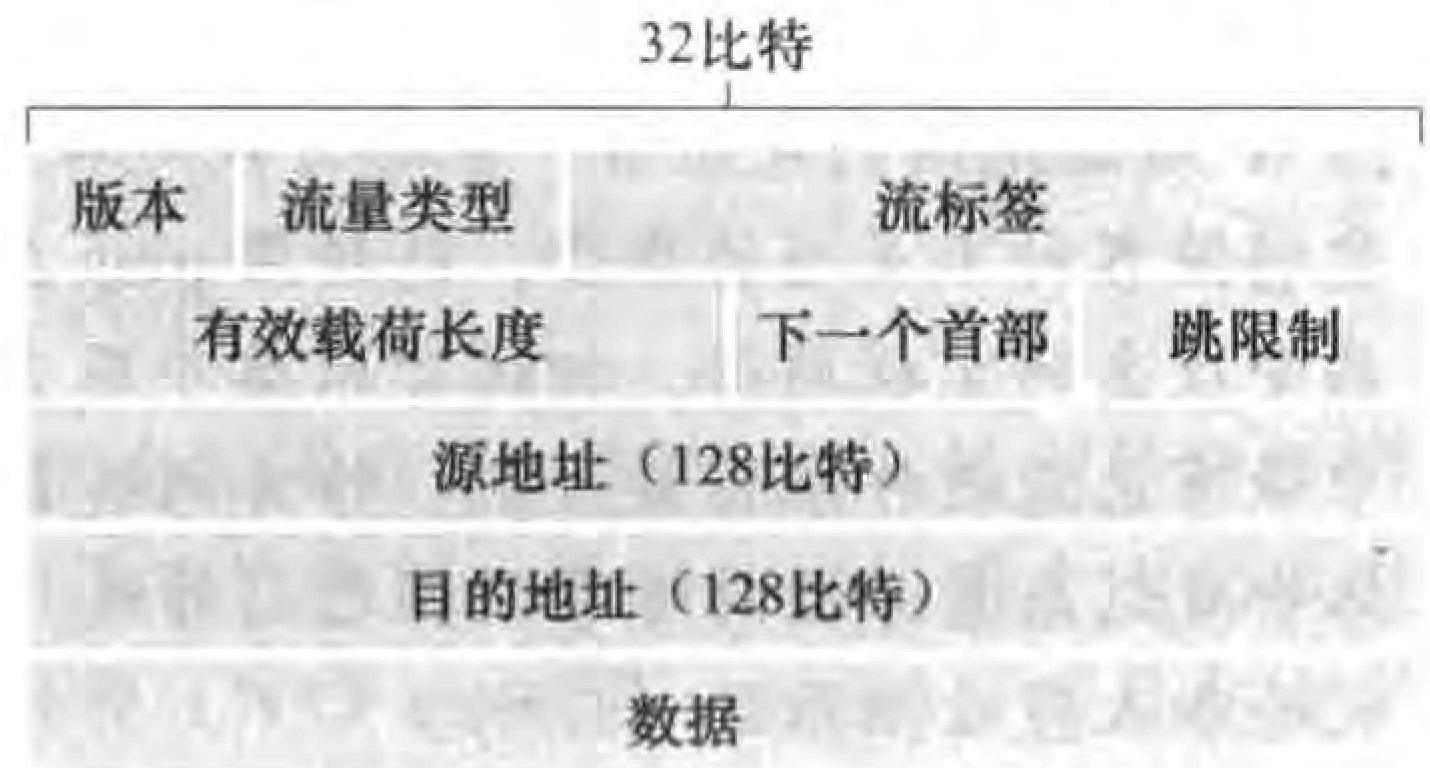


图 4-26　IPv6 数据报格式

1. IPv6 数据报格式

IPv6 数据报的格式如图 4-26 所示。

IPv6 中引入的最重要的变化显示在其数据报格式中：

- 扩大的地址容量。IPv6 将 IP 地址长度从 32 比特增加到 128 比特。这就确保全世界将不会用尽 IP 地址。现在，地球上的每个沙砾都可以用 IP 地址寻址了。除了单播与多播地址以外，IPv6 还引入了一种称为**任播地址**（anycast address）的新型地址，这种地址可以使数据报交付给一组主机中的任意一个。（例如，这种特性可用于向一组包含给定文档的镜像站点中的最近一个发送 HTTP GET 报文。）
- 简化高效的 40 字节首部。如下面讨论的那样，许多 IPv4 字段已被舍弃或作为选项。因而所形成的 40 字节定长首部允许路由器更快地处理 IP 数据报。一种新的选项编码允许进行更灵活的选项处理。
- 流标签。IPv6 有一个难以捉摸的**流**（flow）定义。RFC 2460 中描述道，该字段可用于“给属于特殊流的分组加上标签，这些特殊流是发送方要求进行特殊处理的流，如一种非默认服务质量或需要实时服务的流”。例如，音频与视频传输就可能被当作一个流。另一方面，更为传统的应用（如文件传输和电子邮件）就不可能被当作流。由高优先权用户（如某些为使其流量得到更好服务而付费的用户）承载的流量也有可能被当作一个流。然而，IPv6 的设计者显然已预见到最终需要能够区分这些流，即使流的确切含义还未完全确定。

如上所述，比较图 4-26 与图 4-16 就可看出，IPv6 数据报的结构更简单、更高效。以下是在 IPv6 中定义的字段。

- 版本。该 4 比特字段用于标识 IP 版本号。毫不奇怪，IPv6 将该字段值设为 6。注意到将该字段值置为 4 并不能创建一个合法的 IPv4 数据报。（如果这样的话，事情就简单多了，参见下面有关从 IPv4 向 IPv6 迁移的讨论。）
- 流量类型。该 8 比特字段与我们在 IPv4 中看到的 TOS 字段的含义相似。
- 流标签。如上面讨论过的那样，该 20 比特的字段用于标识一条数据报的流，能够

对一条流中的某些数据报给出优先权，或者它能够用来对来自某些应用（例如 IP 话音）的数据报给出更高的优先权，以优于来自其他应用（例如 SMTP 电子邮件）的数据报。

- *有效载荷长度*。该 16 比特值作为一个无符号整数，给出了 IPv6 数据报中跟在定长的 40 字节数据报首部后面的字节数量。
- *下一个首部*。该字段标识数据报中的内容（数据字段）需要交付给哪个协议（如 TCP 或 UDP）。该字段使用与 IPv4 首部中协议字段相同的值。
- *跳限制*。转发数据报的每台路由器将对该字段的内容减 1。如果跳限制计数达到 0，则该数据报将被丢弃。
- *源地址和目的地址*。IPv6 128 比特地址的各种格式在 RFC 4291 中进行了描述。
- *数据*。这是 IPv6 数据报的有效载荷部分。当数据报到达目的地时，该有效载荷就从 IP 数据报中移出，并交给在下一个首部字段中指定的协议处理。

以上讨论说明了 IPv6 数据报中包括的各字段的用途。将图 4-26 中的 IPv6 数据报格式与图 4-16 中的 IPv4 数据报格式进行比较，我们就会注意到，在 IPv4 数据报中出现的几个字段在 IPv6 数据报中已不复存在：

- *分片/重新组装*。IPv6 不允许在中间路由器上进行分片与重新组装。这种操作只能在源与目的地执行。如果路由器收到的 IPv6 数据报因太大而不能转发到出链路上的话，则路由器只需丢掉该数据报，并向发送方发回一个“分组太大”的 ICMP 差错报文即可（见 5.6 节）。于是发送方能够使用较小长度的 IP 数据报重发数据。分片与重新组装是一个耗时的操作，将该功能从路由器中删除并放到端系统中，大大加快了网络中的 IP 转发速度。
- *首部检验和*。因为因特网层中的运输层（如 TCP 与 UDP）和数据链路层（如以太网）协议执行了检验操作，IP 设计者大概觉得在网络层中具有该项功能实属多余，所以将其去除。再次强调的是，快速处理 IP 分组是关注的重点。在 4.3.1 节中我们讨论 IPv4 时讲过，由于 IPv4 首部中包含有一个TTL字段（类似于 IPv6 中的跳限制字段），所以在每台路由器上都需要重新计算 IPv4 首部检验和。就像分片与重新组装一样，在 IPv4 中这也是一项耗时的操作。
- *选项*。选项字段不再是标准 IP 首部的一部分了。但它并没有消失，而是可能出现在 IPv6 首部中由“下一个首部”指出的位置上。这就是说，就像 TCP 或 UDP 协议首部能够是 IP 分组中的“下一个首部”一样，选项字段也能是“下一个首部”。删除选项字段使得 IP 首部成为定长的 40 字节。

2. 从 IPv4 到 IPv6 的迁移

既然我们已了解了 IPv6 的技术细节，那么我们考虑一个非常实际的问题：基于 IPv4 的公共因特网如何迁移到 IPv6 呢？问题是，虽然新型 IPv6 使能系统可做成向后兼容，即能发送、路由和接收 IPv4 数据报，但已部署的具有 IPv4 能力的系统却不能够处理 IPv6 数据报。可以采用以下几种方法 [Huston 2011b；RFC 4213]。

一种可选的方法是宣布一个标志日，即指定某个日期和时间，届时因特网的所有机器都关机并从 IPv4 升级到 IPv6。上次重大的技术迁移（为得到可靠的运输服务，从使用 NCP 迁移到使用 TCP）出现在差不多 35 年以前。即使回到那时 [RFC 801] ——因特网很小且仍然由少数“奇才”管理着，人们也会认识到选择这样一个标志日是不可行的。一

个涉及数十亿台机器的标志日现在更是不可想象的。

在实践中已经得到广泛采用的 IPv4 到 IPv6 迁移的方法包括**建隧道**（tunneling）[RFC 4213]。除了 IPv4 到 IPv6 迁移之外的许多其他场合的应用都具有建隧道的关键概念，包括在第7章将涉及的全 IP 蜂窝网络中也得到广泛使用。建隧道依据的基本思想如下：假定两个 IPv6 节点（如图 4-27 中的 B 和 E）要使用 IPv6 数据报进行交互，但它们是经由中间 IPv4 路由器互联的。我们将两台 IPv6 路由器之间的中间 IPv4 路由器的集合称为一个**隧道**（tunnel），如图 4-27 所示。借助于隧道，在隧道发送端的 IPv6 节点（如 B）可将整个 IPv6 数据报放到一个 IPv4 数据报的数据（有效载荷）字段中。于是，该 IPv4 数据报的地址设为指向隧道接收端的 IPv6 节点（在此例中为 E），再发送给隧道中的第一个节点（在此例中为 C）。隧道中的中间 IPv4 路由器在它们之间为该数据报提供路由，就像对待其他数据报一样，完全不知道该 IPv4 数据报自身就含有一个完整的 IPv6 数据报。隧道接收端的 IPv6 节点最终收到该 IPv4 数据报（它是该 IPv4 数据报的目的地），并确定该 IPv4 数据报含有一个 IPv6 数据报（通过观察在 IPv4 数据报中的协议号字段是 41 [RFC 4213]，指示该 IPv4 有效载荷是 IPv6 数据报），从中取出 IPv6 数据报，然后再为该 IPv6 数据报提供路由，就好像它是从一个直接相连的 IPv6 邻居那里接收到该 IPv6 数据报一样。

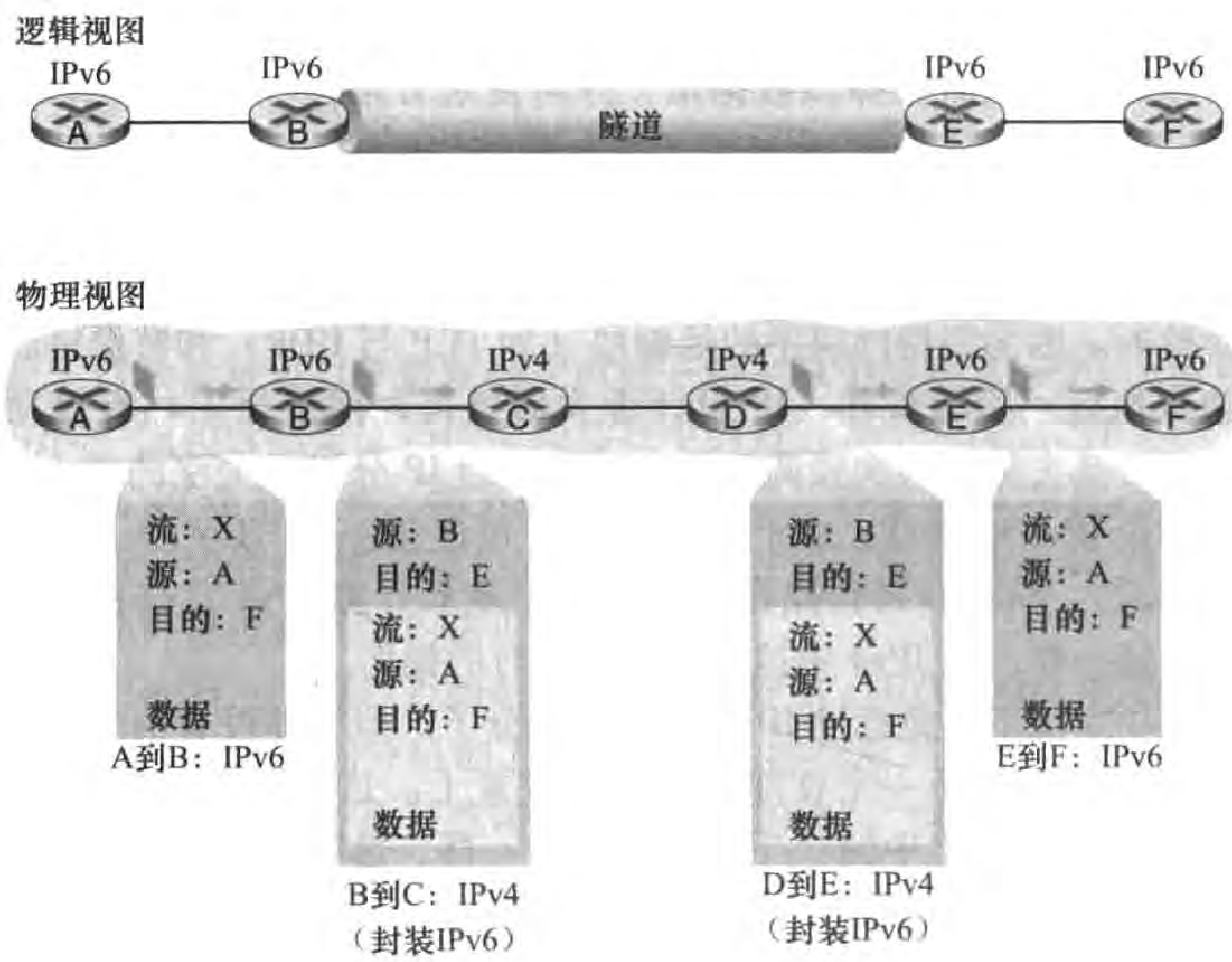


图 4-27 建隧道

在结束本节前需要说明的是，尽管采用 IPv6 最初表现为一个缓慢启动的过程 [Lawton 2001；Huston 2008b]，但势头已经有了。NIST [NIST IPv6 2015] 报告称，超过三分之一的美国政府二级域名是支持 IPv6 的。在客户端，谷歌报告称访问谷歌服务的客户仅有 8% 使用了 IPv6 [Google IPv6 2015]。但其他最近统计结果指出 [Czyz 2014]，IPv6 的采用正在加速。诸如 IP 使能的电话和其他便携式设备的激增，为 IPv6 的更广泛部署提供了新的推动力。欧洲的第三代合作计划 [3GPP 2016] 已规定了 IPv6 为移动多媒体的标准编址方案。

我们能从IPv6经验中学到的重要一课是，要改变网络层协议是极其困难的。自从20世纪90年代早期以来，有许多新的网络层协议被鼓吹为因特网的下一次重大革命，但这些协议中的大多数至今为止只取得了有限突破。这些协议包括IPv6、多播协议、资源预留协议，其中后面两个协议的讨论可在本书的在线补充材料中找到。在网络层中引入新的协议的确如同替换一幢房子的基石，即在不拆掉整幢房子（或至少临时重新安置房屋住户）的情况下是很难完成上述工作的。另一方面，因特网却已见证了在应用层中新协议的快速部署。典型的例子当然有Web、即时讯息、流媒体、分布式游戏和各种形式的社交媒体。引入新的应用层协议就像给一幢房子重新刷一层漆，这是相对容易做的事，如果你选择了一个好看的颜色，邻居将会照搬你的选择。总之，未来我们肯定会看到因特网网络层发生改变，但这种改变将比应用层慢得多。

4.4 通用转发和SDN

在4.2.1节中，我们注意到因特网路由器的转发决定传统上仅仅基于分组的目的地址。然而，在前一节中，我们也已经看到执行许多第三层功能的中间盒有了大量发展。NAT盒重写首部IP地址和端口号；防火墙基于首部字段值阻拦流量或重定向分组以进行其他处理，如深度分组检测（DPI）。负载均衡器将请求某种给定服务（例如一个HTTP请求）的分组转发到提供该服务的服务器集合中的一个。[RFC 3234] 列出了许多常用中间盒功能。

第二层交换机和第三层路由器等中间盒 [Qazi 2013] 的剧增，而且每种都有自己特殊的硬件、软件和管理界面，无疑给许多网络操作员带来了十分头疼的大麻烦。然而，近期软件定义网络的进展已经预示并且正在提出一种统一的方法，以一种现代、简洁和综合方式，提供多种网络层功能以及某些链路层功能。

回顾4.2.1节将基于目的地转发的特征总结为两个步骤：查找目的IP地址（“匹配”），然后将分组发送到有特定输出端口的交换结构（“动作”）。我们现在考虑一种更有意义的通用“匹配加动作”范式，其中能够对协议栈的多个首部字段进行“匹配”，这些首部字段是与不同层次的不同协议相关联的。“动作”能够包括：将分组转发到一个或多个输出端口（就像在基于目的地转发中一样），跨越多个通向服务的离开接口进行负载均衡分组（就像在负载均衡中一样），重写首部值（就像在NAT中一样），有意识地阻挡/丢弃某个分组（就像在防火墙中一样），为进一步处理和动作而向某个特定的服务器发送一个分组（就像在DPI一样），等等。

在通用转发中，一张匹配加动作表将我们在4.2.1节中看到的基于目的地的转发表一般化了。因为能够使用网络层和/或链路层源和目的地址做出转发决定，所以显示在图4-28中的转发设备更为准确地描述为“分组交换机”而不是第三层“路由器”或第二层“交换机”。因此，在本节后面部分以及5.5节中，我们将这些设备称为分组交换机，这是在SDN文献中被广泛采用的术语。

图4-28显示了位于每台分组交换机中的一张匹配加动作表，该表由远程控制器计算、安装和更新。我们注意到虽然在各台分组交换机中的控制组件可以相互作用（例如以类似于图4-2中的方式），但实践中通用匹配加动作能力是通过计算、安装和更新这些表的远程控制器实现的。花几分钟比较图4-2、图4-3和图4-28，你能看出图4-2和图4-3中显示的基于目的地转发与图4-28中显示的通用转发有什么相似和差异吗？

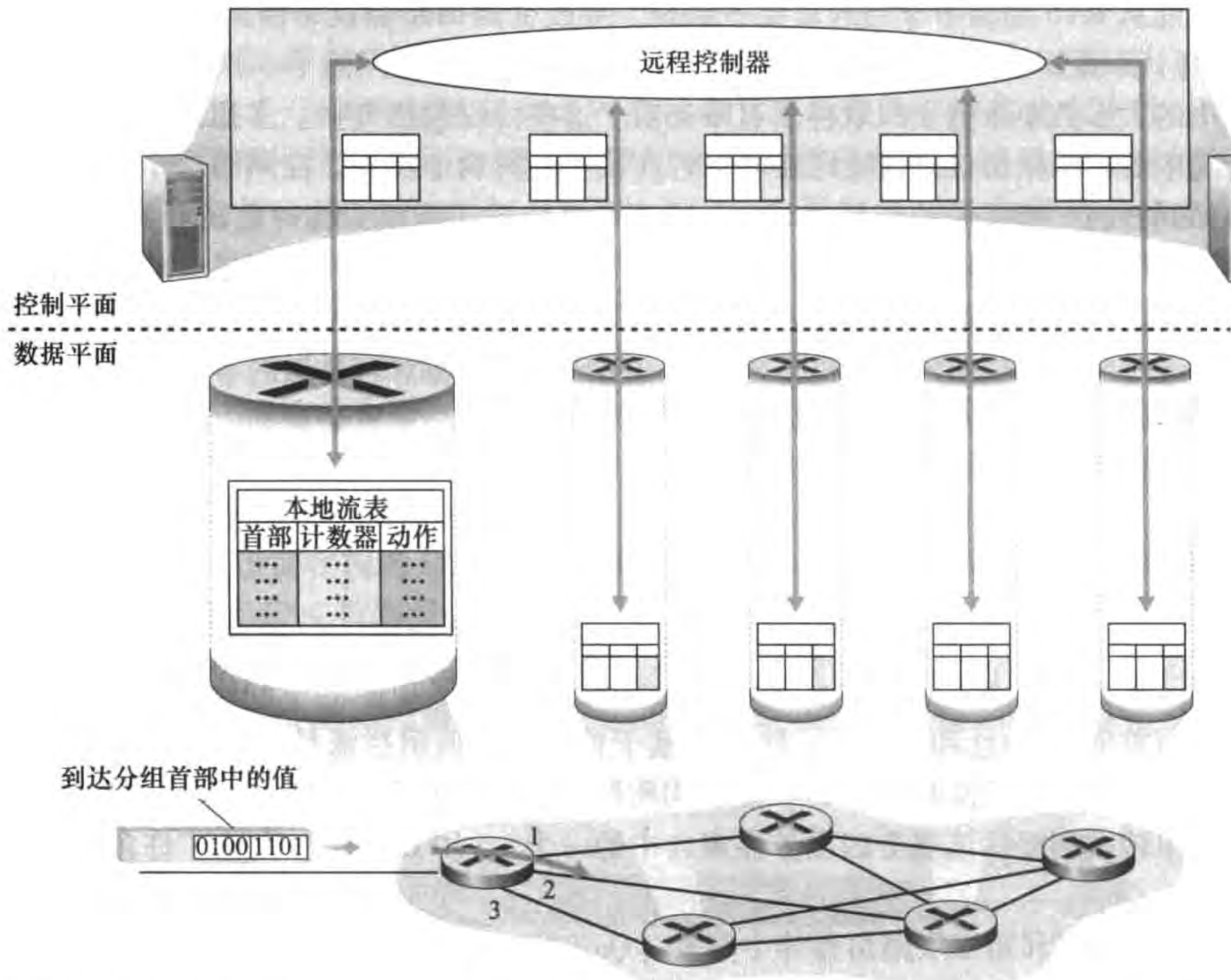


图 4-28 通用转发：每台分组交换机包含一张匹配加动作表，该表是由远程控制器计算和分发的

我们后续对通用转发的讨论将基于 OpenFlow [McKeown 2008；OpenFlow 2009；Casado 2014；Tourrilhes 2014]，OpenFlow 是一个得到高度认可和成功的标准，它已经成为匹配加动作转发抽象、控制器以及更为一般的 SDN 革命等概念的先驱 [Feamster 2013]。我们将主要考虑 OpenFlow 1.0，该标准以特别清晰和简明的方式引入了关键的 SDN 抽象和功能。OpenFlow 的后继版本根据实现和使用获得的经验引入了其他能力；OpenFlow 标准的当前和早期版本能在 [ONF 2016] 中找到。

匹配加动作转发表在 OpenFlow 中称为**流表**（flow table），它的每个表项包括：

- 首部字段值的集合，入分组将与之匹配。与基于目的地转发的情况一样，基于硬件匹配在 TCAM 内存中执行得最为迅速（TCAM 内存中可能有上百万条地址表项）[Bosshart 2013]。匹配不上流表项的分组将被丢弃或发送到远程控制器做更多处理。在实践中，为了性能或成本原因，一个流表可以由多个流表实现 [Bosshart 2013]，但我们这里只关注单一流表的抽象。
- 计数器集合（当分组与流表项匹配时更新计数器）。这些计数器可以包括已经与该表项匹配的分组数量，以及自从该表项上次更新以来的时间。
- 当分组匹配流表项时所采取的动作集合。这些动作可能将分组转发到给定的输出端口，丢弃该分组、复制该分组和将它们发送到多个输出端口，和/或重写所选的首部字段。

我们将在 4.4.1 节和 4.4.2 节中分别更为详细地探讨匹配和动作。我们将学习每台分组交换机网络范围的匹配规则集合是如何用来实现多种多样的功能的，包括 4.4.3 节中的

路由选择、第二层交换路由、防火墙、负载均衡、虚拟网络等等。在结束时，我们注意到流表本质上是一个 API，通过这种抽象每台分组交换机的行为能被编程；我们将在 4.4.3 节中看到，通过在网络分组交换机的集合中适当地编程/配置这些表，网络范围的行为能被类似地编程［Casado 2014］。

4.4.1 匹配

图 4-29 显示了 11 个分组首部字段和入端口 ID，该 ID 能被 OpenFlow 1.0 中的匹配加动作规则所匹配。前面 1.5.2 节讲过，到达一台分组交换机的一个链路层（第二层）帧将包含一个网络层（第三层）数据报作为其有效载荷，该载荷通常依次将包含一个运输层（第四层）报文段。第一个观察是，OpenFlow 的匹配抽象允许对来自三个层次的协议首部所选择的字段进行匹配（因此相当勇敢地违反了我们在 1.5 节中学习的分层原则）。因为我们还没有涉及链路层，用如下的说法也就足够了：显示在图 4-29 中的源和目的 MAC 地址是与帧的发送和接收接口相关联的链路层地址；通过基于以太网地址而不是 IP 地址进行转发，我们看到 OpenFlow 使能的设备能够等价于路由器（第三层设备）转发数据报以及交换机（第二层设备）转发帧。以太网类型字段对应于较高层协议（例如 IP），利用该字段分解该帧的载荷，并且 VLAN 字段与所谓虚拟局域网相关联，我们将在第 6 章中学习 VLAN。OpenFlow 1.0 规范中匹配的 12 个值在最近的 OpenFlow 规范中已经增加到 41 个［Bosshart 2014］。



图 4-29　OpenFlow 1.0 流表的分组匹配字段

入端口是指分组交换机上接收分组的输入端口。在 4.3.1 节中，我们已经讨论过该分组的 IP 源地址、IP 目的地址、IP 协议字段和 IP 服务类型字段。运输层源和目的端口号字段也能匹配。

流表项也可以有通配符。例如，在一个流表中 IP 地址 128.119.＊.＊将匹配其地址的前 16 比特为 128.119 的任何数据报所对应的地址字段。每个流表项也具有相应的优先权。如果一个分组匹配多个流表项，选定的匹配和对应的动作将是其中有最高优先权的那个。

最后，我们观察到并非一个 IP 首部中的所有字段都能被匹配。例如 OpenFlow 并不允许基于 TTL 字段或数据报长度字段的匹配。为什么有些字段允许匹配，而有些字段不允许呢？毫无疑问，与功能和复杂性有关。选择一种抽象的“艺术”是提供足够的功能来完成某种任务（在这种情况下是实现、配置和管理宽泛的网络层功能，以前这些一直是通过各种各样的网络层设备来实现的），不必用如此详尽和一般性的“超负荷”抽象，这种抽象已经变得臃肿和不可用。Butler Lampson 有过著名的论述［Lampson 1983］：

在一个时刻做一件事，将它做好。一个接口应当俘获一个抽象的最低限度的要件。不要进行一般化，一般化通常是错误的。

考虑到 OpenFlow 的成功，人们能够推测它的设计者的确很好地选择了抽象技术。OpenFlow 匹配的更多细节能够在［OpenFlow 2009；ONF 2016］中找到。

4.4.2 动作

如图 4-28 中所见，每个流表项都有零个或多个动作列表，这些动作决定了应用于与流表项匹配的分组的处理。如果有多个动作，它们以在表中规定的次序执行。

其中最为重要的动作可能是：

- *转发*。一个入分组可以转发到一个特定的物理输出端口，广播到所有端口（分组到达的端口除外），或通过所选的端口集合进行多播。该分组可能被封装并发送到用于该设备的远程控制器。该控制器则可能（或可能不）对该分组采取某些动作，包括安装新的流表项，以及可能将该分组返回给该设备以在更新的流表规则集合下进行转发。
- *丢弃*。没有动作的流表项表明某个匹配的分组应当被丢弃。
- *修改字段*。在分组被转发到所选的输出端口之前，分组首部 10 个字段（图 4-29 中显示的除 IP 协议字段外的所有第二、三、四层的字段）中的值可以重写。

4.4.3 匹配加动作操作中的 OpenFlow 例子

在已经考虑了通用转发的匹配和动作组件后，我们在图 4-30 显示的样本网络场景中将这些想法拼装在一起。该网络具有 6 台主机（h1、h2、h3、h4、h5 和 h6）以及 3 台分组交换机（s1、s2 和 s3），每台交换机具有 4 个本地接口（编号 1 到 4）。我们将考虑一些希望实现的网络范围的行为，在 s1、s2 和 s3 中的流表项需要实现这种行为。

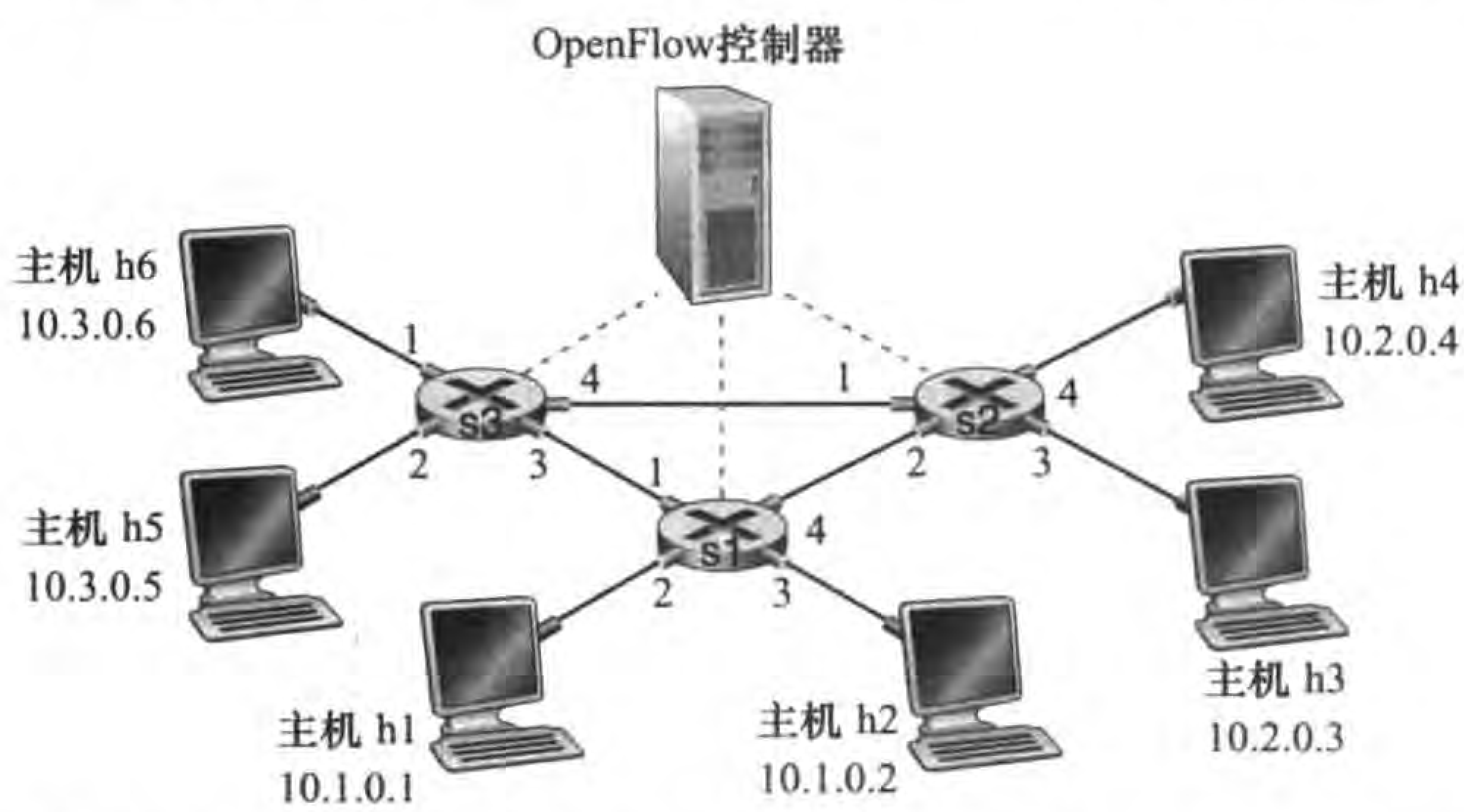


图 4-30 具有 3 台分组交换机、6 台主机和 1 台 OpenFlow 控制器的 OpenFlow 匹配加动作网络

第一个例子：简单转发

作为一个非常简单的例子，假定希望的转发行为是：来自 h5 或 h6 发往 h3 或 h4 的分组从 s3 转发到 s1，然后从 s1 转发到 s2（完全避免使用 s3 和 s2 之间的链路）。在 s1 中的流表项将是：

s1 流表（例 1）	
匹配	动作
Ingress Port = 1; IP Src = 10.3.*.*; IP Dst = 10.2.*.*	Forward (4)
……	……

当然，我们也需要在 s3 中有一个流表项，使得该数据报从 h5 或 h6 经过出接口 3 转发到 s1：

s3 流表（例 1）	
匹配	动作
IP Src = 10.3.*.*；IP Dst = 10.2.*.*	Forward (3)
……	……

最后，我们也需要在 s2 中有一个流表项来完成第一个例子，使得从 s1 到达的数据报转发到它们的目的主机 h3 或 h4。

s2 流表（例 1）	
匹配	动作
Ingress Port = 2；IP Dst = 10.2.0.3	Forward (3)
Ingress Port = 2；IP Dst = 10.2.0.4	Forward (4)
……	……

第二个例子：负载均衡

作为第二个例子，我们考虑一个负载均衡的场景，其中来自 h3 发往 10.1.*.* 的数据报经过 s1 和 s2 之间的直接链路转发，与此同时来自 h4 发往 10.1.*.* 的数据报经过 s2 和 s3（于是从 s3 到 s1）之间的链路转发。注意到这种行为不能通过基于 IP 的目的地转发取得。在这种情况下，在 s2 中的流表项将是：

s2 流表（例 2）	
匹配	动作
Ingress Port = 3；IP Dst = 10.1.*.*	Forward (2)
Ingress Port = 4；IP Dst = 10.1.*.*	Forward (1)
……	……

在 s1 中需要流表项将从 s2 收到的数据报转发到 h1 或 h2；在 s3 中需要流表项将接口 4 上从 s2 收到的数据报经过接口 3 转发到 s1。考虑是否能在 s1 和 s3 中配置这些流表项。

第三个例子：充当防火墙

作为第三个例子，我们考虑一个防火墙场景，其中 s2 仅希望（在它的任何接口上）接收来自与 s3 相连的主机所发送的流量。

s2 流表（例 3）	
匹配	动作
IP Src = 10.3.*.*；IP Dst = 10.2.0.3	Forward (3)
IP Src = 10.3.*.*；IP Dst = 10.2.0.4	Forward (4)
……	……

如果在 s2 的流表中没有其他表项，则仅有来自 10.3.*.* 的流量将被转发到与 s2 相连的主机。

尽管我们这里仅考虑了几种基本场景，但通用转发的多样性和优势显而易见。在课后习题中，我们将探讨流表如何用来生成许多不同的逻辑行为，包括使用相同分组交换机和

链路物理集合的虚拟网络，即两个或多个逻辑上分离的网络（每个网络有它们自己的独立和截然不同的转发行为）。在 5.5 节中，当学习 SDN 控制器时，我们将再次考察流表，其中 SDN 控制器计算和分发流表，协议用于在分组交换机和它的控制器之间进行通信。

4.5 小结

在本章中，我们讨论了网络层的**数据平面**（data plane）功能，即每台路由器的如下功能：决定到达路由器的输入链路之一的分组如何转发到该路由器的输出链路之一。

我们从仔细观察路由器的内部操作开始，学习输入和输出端口功能，以及基于目的地的转发、路由器的内部交换机制、分组排队管理等等。我们涉及传统的 IP 转发（其中转发基于数据报的目的地址进行）和通用转发（其中转发和其他功能可以使用数据报首部中的几个不同的字段值来进行），并且看到了后一种方法的多种用途。我们还详细地学习了 IPv4 和 IPv6 协议以及因特网编址，并对此有了更深入、更敏锐和更有趣的发现。

借助于新得到的对网络层数据平面的理解，我们现在准备着手学习第 5 章中的网络层控制平面。

课后习题和问题

复习题

4.1 节

R1. 我们回顾在本书中使用的某些术语。前面讲过运输层的分组名字是报文段，数据链路层的分组名字是帧。网络层的分组名字是什么？前面讲过路由器和链路层交换机都被称为分组交换机。路由器与链路层交换机间的根本区别是什么？

R2. 我们注意到网络层功能可被大体分成数据平面功能和控制平面功能。数据平面的主要功能是什么？控制平面的主要功能呢？

R3. 我们对网络层执行的转发功能和路由选择功能进行区别。路由选择和转发的主要区别是什么？

R4. 路由器中转发表的主要作用是什么？

R5. 我们说过网络层的服务模型“定义发送主机和接收主机之间端到端分组的传送特性”。因特网的网络层的服务模型是什么？就主机到主机数据报的传递而论，因特网的服务模型能够保证什么？

4.2 节

R6. 在 4.2 节中，我们看到路由器通常由输入端口、输出端口、交换结构和路由选择处理器组成。其中哪些是用硬件实现的，哪些是用软件实现的？为什么？转到网络层的数据平面和控制平面的概念，哪些是用硬件实现的，哪些是用软件实现的？为什么？

R7. 讨论为什么在高速路由器的每个输入端口都存储转发表的影子副本。

R8. 基于目的地转发意味着什么？这与通用转发有什么不同（假定你已经阅读 4.4 节，两种方法中哪种是软件定义网络所采用的）？

R9. 假设一个到达分组匹配了路由器转发表中的两个或更多表项。采用传统的基于目的地转发，路由器用什么原则来确定这条规则可以用于确定输出端口，使得到达的分组能交换到输出端口？

R10. 在 4.2 节中讨论了三种交换结构。列出并简要讨论每一种交换结构。哪一种（如果有的话）能够跨越交换结构并行发送多个分组？

R11. 描述在输入端口会出现分组丢失的原因。描述在输入端口如何消除分组丢失（不使用无限大缓存区）。

R12. 描述在输出端口会出现分组丢失的原因。通过提高交换结构速率，能够防止这种丢失吗？

R13. 什么是 HOL 阻塞？它出现在输入端口还是输出端口？

R14. 在4.2节我们学习了FIFO、优先权、循环（RR）和加权公平排队（WFQ）分组调度规则。这些排队规则中，哪个规则确保所有分组是以到达的次序离开的？

R15. 举例说明为什么网络操作员要让一类分组的优先权超过另一类分组的。

R16. RR和WFQ分组调度之间的基本差异是什么？存在RR和WFQ将表现得完全相同的场合吗？（提示：考虑WFQ权重。）

4.3节

R17. 假定主机A向主机B发送封装在一个IP数据报中的TCP报文段。当主机B接收到该数据报时，主机B中的网络层怎样知道它应当将该报文段（即数据报的有效载荷）交给TCP而不是UDP或某个其他东西呢？

R18. 在IP首部中，哪个字段能用来确保一个分组的转发不超过N台路由器？

R19. 前面讲过因特网检验和被用于运输层报文段（分别在图3-7和图3-29的UDP和TCP首部中）以及网络层数据报（图4-16的IP首部中）。现在考虑一个运输层报文段封装在一个IP数据报中。在报文段首部和数据报首部中的检验和要遍及IP数据报中的任何共同字节进行计算吗？

R20. 什么时候一个大数据报分割成多个较小的数据报？较小的数据报在什么地方装配成一个较大的数据报？

R21. 路由器有IP地址吗？如果有，有多少个？

R22. IP地址223.1.3.27的32比特二进制等价形式是什么？

R23. 考察使用DHCP的主机，获取它的IP地址、网络掩码、默认路由器及其本地DNS服务器的IP地址。列出这些值。

R24. 假设在一个源主机和一个目的主机之间有3台路由器。不考虑分片，一个从源主机发送给目的主机的IP数据报将通过多少个接口？为了将数据报从源移动到目的地需要检索多少个转发表？

R25. 假设某应用每20ms生成一个40字节的数据块，每块封装在一个TCP报文段中，TCP报文段再封装在一个IP数据报中。每个数据报的开销有多大？应用数据所占百分比是多少？

R26. 假定你购买了一个无线路由器并将其与电缆调制解调器相连。同时假定ISP动态地为你连接的设备（即你的无线路由器）分配一个IP地址。还假定你家有5台PC，均使用802.11以无线方式与该无线路由器相连。怎样为这5台PC分配IP地址？该无线路由器使用NAT吗？为什么？

R27. “路由聚合”一词意味着什么？路由器执行路由聚合为什么是有用的？

R28. “即插即用”或“零配置”协议意味着什么？

R29. 什么是专用网络地址？具有专用网络地址的数据报会出现在大型公共因特网中吗？解释理由。

R30. 比较并对照IPv4和IPv6首部字段。它们有相同的字段吗？

R31. 有人说当IPv6以隧道形式通过IPv4路由器时，IPv6将IPv4隧道作为链路层协议。你同意这种说法吗？为什么？

4.4节

R32. 通用转发与基于目的地转发有何不同？

R33. 我们在4.1节遇到的基于目的地转发与在4.4节遇到的OpenFlow流表之间有什么差异？

R34. 路由器或交换机的“匹配加动作”意味着什么？在基于目的地转发的分组交换机场合中，要匹配什么并采取什么动作？在SDN的场合中，举出3个能够被匹配的字段和3个能被采取的动作。

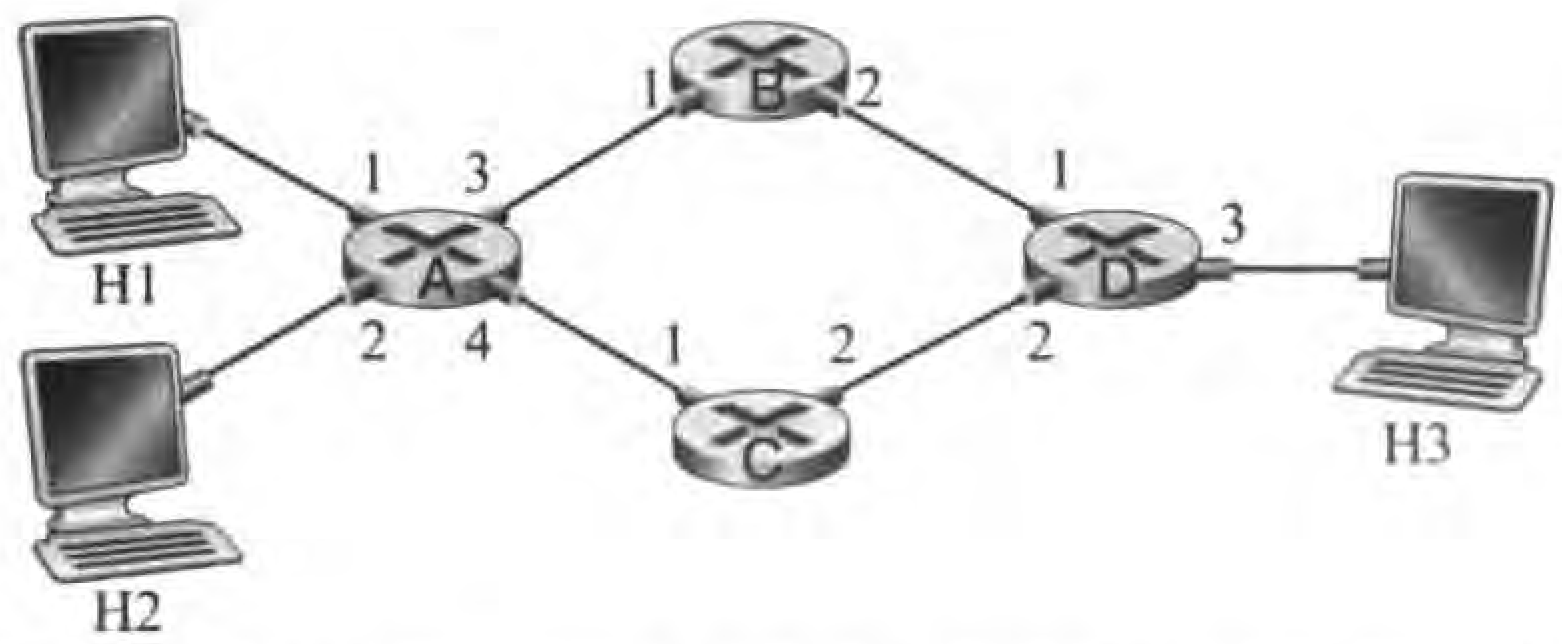
R35. 在IP数据报中举出能够在OpenFlow 1.0通用转发中“匹配”的3个首部字段。不能在OpenFlow中“匹配”的3个IP数据报首部字段是什么？

习题

P1. 考虑下面的网络。

a. 显示路由器A中的转发表，使得目的地为主机H3的所有流量都通过接口3转发。

b. 写出路由器A中的转发表，使得从H1发往主机H3的所有流量都通过接口3转发，从H2发往主机H3的所有流量都通过接口4转发。（提示：这是一个技巧性的问题。）



P2. 假设两个分组在完全相同的时刻到达一台路由器的两个不同输入端口。同时假设在该路由器中没有其他分组。

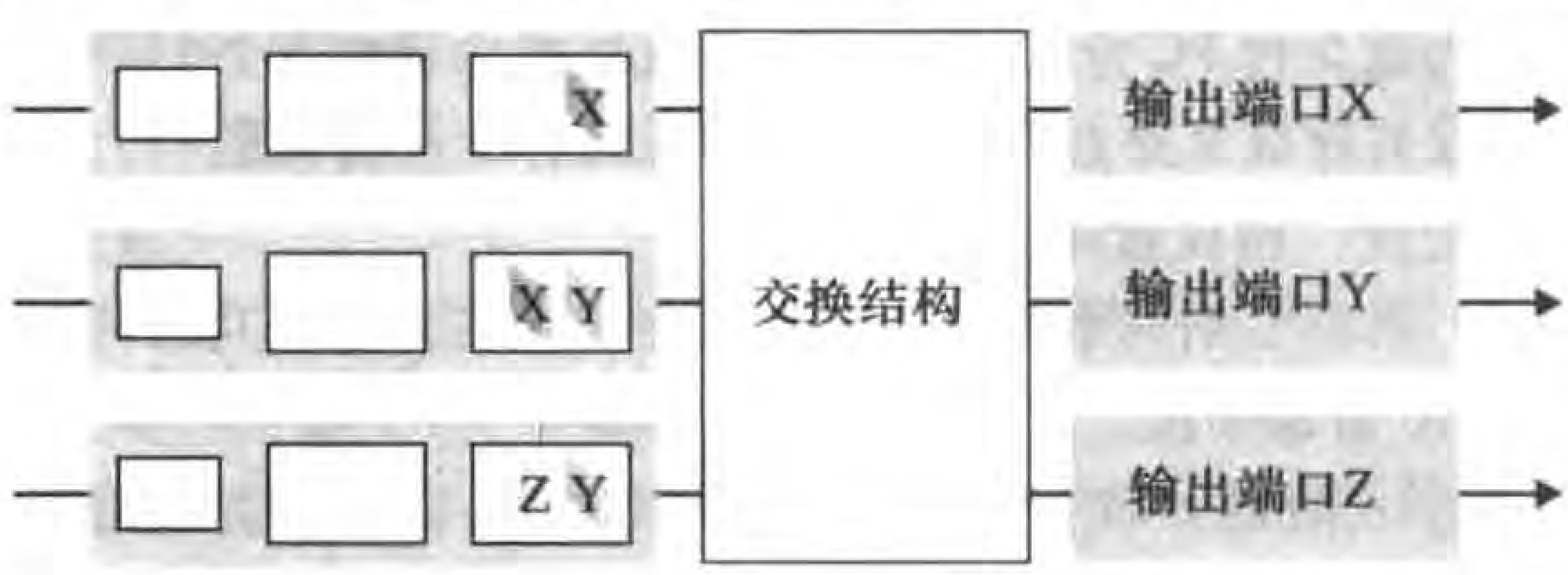
a. 假设这两个分组朝着两个不同的输出端口转发。当交换结构使用一条共享总线时，这两个分组可能在相同时刻通过该交换结构转发吗？

b. 假设这两个分组朝着两个不同的输出端口转发。当交换结构使用经内存交换时，这两个分组可能在相同时刻通过该交换结构转发吗？

c. 假设这两个分组朝着相同的输出端口转发。当交换结构使用纵横式时，这两个分组可能在相同时刻通过该交换结构转发吗？

P3. 在4.2节中，我们注意到如果交换结构速率是输入线路速率的 n 倍，其最大的排队时延为 $(n-1)D$。假设所有分组有相同长度，在相同时刻 n 个分组到达 n 个输出端口，同时所有 n 个分组要转发到不同的输出端口。对于内存、总线和纵横式交换结构，一个分组的最大时延是多少？

P4. 考虑下列交换机。假设所有数据报具有相同长度，交换机以一种分时隙、同步的方式运行，在一个时隙中一个数据报能够从某输入端口传送到某输出端口。其交换结构是纵横式的，因此在一个时隙中至多一个数据报能够传送到一个给定输出端口，但在一个时隙中不同的输出端口能够接收到来自不同输入端口的数据报，从输入端口到它们的输出端口传送所示的分组，所需的时隙数量最小是多少？此时假定使用你所需要的任何输入排队调度方法（即此时没有HOL阻塞）。假定采用你能够设计的最差情况下的调度方案，且非空输入队列不会空闲，所需的时隙数量最大是多少？



P5. 考虑使用32比特主机地址的某数据报网络。假定一台路由器具有4条链路，编号为0～3，分组能被转发到如下的各链路接口：

目的地址范围	链路接口
11100000 00000000 00000000 00000000 到 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 到 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 到 11100001 01111111 11111111 11111111	2
其他	3

a. 提供一个具有 5 个表项的转发表，使用最长前缀匹配，转发分组到正确的链路接口。

b. 描述你的转发表是如何为具有下列目的地址的数据报决定适当的链路接口的。

11001000 10010001 01010001 01010101

11100001 01000000 11000011 00111100

11100001 10000000 00010001 01110111

P6. 考虑使用 8 比特主机地址的某数据报网络。假定一台路由器使用最长前缀匹配并具有下列转发表：

前缀匹配	接口
00	0
010	1
011	2
10	2
11	3

对这 4 个接口中的每个，给出相应的目的主机地址的范围和在该范围中的地址数量。

P7. 考虑使用 8 比特主机地址的数据报网络。假定一台路由器使用最长前缀匹配并具有下列转发表：

前缀匹配	接口
1	0
10	1
111	2
其他	3

对这 4 个接口中的每个，给出相应的目的主机地址的范围和在该范围中的地址数量。

P8. 考虑互联 3 个子网（子网 1、子网 2 和子网 3）的一台路由器。假定这 3 个子网的所有接口要求具有前缀 223. 1. 17/24。还假定子网 1 要求支持多达 60 个接口，子网 2 要求支持多达 90 个接口，子网 3 要求支持多达 12 个接口。提供 3 个满足这些限制的网络地址（形式为 *a. b. c. d/x*）。

P9. 在 4. 2. 2 节中给出了一个转发表（使用最长前缀匹配）的例子。使用 *a. b. c. d/x* 记法代替二进制字符串记法，重写该转发表。

P10. 在习题 P5 中要求你给出转发表（使用最长前缀匹配）。使用 *a. b. c. d/x* 记法代替二进制字符串记法，重写该转发表。

P11. 考虑一个具有前缀 128. 119. 40. 128/26 的子网。给出能被分配给该网络的一个 IP 地址（形式为 xxx. xxx. xxx. xxx）的例子。假定一个 ISP 拥有形式为 128. 119. 40. 64/26 的地址块。假定它要从该地址块生成 4 个子网，每块具有相同数量的 IP 地址。这 4 个子网（形式为 *a. b. c. d/x*）的前缀是什么？

P12. 考虑图 4-20 中显示的拓扑。（在 12:00 以顺时针开始）标记具有主机的 3 个子网为网络 A、B 和 C，标记没有主机的子网为网络 D、E 和 F。

a. 为这 6 个子网分配网络地址，要满足下列限制：所有地址必须从 214. 97. 254/23 起分配；子网 A 应当具有足够地址以支持 250 个接口；子网 B 应当具有足够地址以支持 120 个接口；子网 C 应当具有足够地址以支持 120 个接口。当然，子网 D、E 和 F 应当支持两个接口。对于每个子网，分配采用的形式是 *a. b. c. d/x* 或 *a. b. c. d/x ~ e. f. g. h/y*。

b. 使用你对（a）部分的答案，为这 3 台路由器提供转发表（使用最长前缀匹配）。

P13. 使用美国因特网编码注册机构（http://www. arin. net/whois）的 whois 服务来确定三所大学所用的 IP 地址块。whois 服务能被用于确定某个特定的 IP 地址的确定地理位置吗？使用 www. maxmind. com 来确定位于这三所大学的 Web 服务器的位置。

P14. 考虑向具有 700 字节 MTU 的一条链路发送一个 2400 字节的数据报。假定初始数据报标有标识号 422。将会生成多少个分片？在生成相关分片的数据报中各个字段的值是多少？

P15. 假定在源主机 A 和目的主机 B 之间的数据报被限制为 1500 字节（包括首部）。假设 IP 首部为 20 字节，要发送一个 5MB 的 MP3 文件需要多少个数据报？解释你的答案是如何计算的。

P16. 考虑在图 4-25 中建立的网络。假定 ISP 现在为路由器分配地址 24.34.112.235，家庭网络的网络地址是 192.168.1/24。

a. 在家庭网络中为所有接口分配地址。

b. 假定每台主机具有两个进行中的 TCP 连接，所有都是针对主机 128.119.40.86 的 80 端口的。在 NAT 转换表中提供 6 个对应表项。

P17. 假设你有兴趣检测 NAT 后面的主机数量。你观察到在每个 IP 分组上 IP 层顺序地标出一个标识号。由一台主机生成的第一个 IP 分组的标识号是一个随机数，后继 IP 分组的标识号是顺序分配的。假设由 NAT 后面主机产生的所有 IP 分组都发往外部。

a. 基于这个观察，假定你能够俘获由 NAT 向外部发送的所有分组，你能概要给出一种简单的技术来检测 NAT 后面不同主机的数量吗？评估你的答案。

b. 如果标识号不是顺序分配而是随机分配的，这种技术还能正常工作吗？评估你的答案。

P18. 在这个习题中，我们将探讨 NAT 对 P2P 应用程序的影响。假定用户名为 Arnold 的对等方通过查询发现，用户名为 Bernard 的对等方有一个要下载的文件。同时假定 Bernard 和 Arnold 都位于 NAT 后面。尝试设计一种技术，使得 Arnold 与 Bernard 创建一条 TCP 连接，而不对 NAT 做应用特定的配置。如果难以设计这样的技术，试讨论其原因。

P19. 考虑显示在图 4-30 中的 SDN OpenFlow 网络。假定对于到达 s2 的数据报的期望转发行为如下：

- 来自主机 h5 或 h6 并且发往主机 h1 或 h2 的任何数据报应当通过输出端口 2 转发到输入端口 1。
- 来自主机 h1 或 h2 并且发往主机 h5 或 h6 的任何数据报应当通过输出端口 1 转发到输入端口 2。
- 任何在端口 1 或 2 到达并且发往主机 h3 或 h4 的数据报应当传递到特定的主机。
- 主机 h3 和 h4 应当能够向彼此发送数据报。

详述实现这种转发行为的 s2 中的流表项。

P20. 再次考虑显示在图 4-30 中的 SDN OpenFlow 网络。假定在 s2 对于来自主机 h3 或 h4 的数据报的期望转发行为如下：

- 任何来自主机 h3 并且发往主机 h1、h2、h5 或 h6 的数据报应当在网络中以顺时针方向转发。
- 任何来自主机 h4 并且发往主机 h1、h2、h5 或 h6 的数据报应当在网络中以逆时针方向转发。

详述实现这种转发行为的 s2 中的流表项。

P21. 再次考虑上面 P19 的场景。给出分组交换机 s1 和 s3 的流表项，使得具有 h3 或 h4 源地址的任何到达数据报被路由到在 IP 数据报的目的地址字段中定义的目的主机。（提示：你的转发表规则应当包括如下情况，即到达的数据报被发往直接连接的主机，或应当转发到相邻路由器以便传递到最终主机。）

P22. 再次考虑显示在图 4-30 中的 SDN OpenFlow 网络。假定我们希望交换机 s2 的功能像防火墙一样。在 s2 中定义实现下列防火墙行为的流表，以传递目的地为 h3 和 h4 的数据报（对下列四种防火墙行为，每种定义一张不同的流表）。不需要在 s2 中定义将流量转发到其他路由器的转发行为。

- 仅有从主机 h1 和 h6 到达的流量应当传递到主机 h3 或 h4（即从主机 h2 和 h5 到达的流量被阻塞）。
- 仅有 TCP 流量被允许传递给主机 h3 或 h4（即 UDP 流量被阻塞）。
- 仅有发往 h3 的流量被传递（即所有到 h4 的流量被阻塞）。
- 仅有来自 h1 并且发往 h3 的 UDP 流量被传递。所有其他流量被阻塞。

Wireshark 实验

在与本书配套的 Web 站点 www.pearsonhighered.com/cs-resources 上，你将找到一个 Wireshark 实验作业，该作业考察了 IP 协议的运行，特别是 IP 数据报的格式。

人物专访

Vinton G. Cerf 是 Internet Evangelist for Google 公司的副总裁兼主席。他在 MCI 公司服务了 16 年，担任过各种职位，最后以技术战略部资深副总裁的身份结束了他在那里的任期。他作为 TCP/IP 协议和因特网体系结构的共同设计者而广为人知。1976 年到 1982 年在美国国防部高级研究计划署（DARPA）任职期间，他在领导因特网以及与因特网相关的数据分组和安全技术的研发方面发挥了重要作用。他于 2005 年获得了美国总统自由奖章，于 1977 年获得了美国国家技术奖章。他在斯坦福大学获得数学学士学位，在加利福尼亚大学洛杉矶分校（UCLA）获得了计算机科学的硕士和博士学位。



Vinton G. Cerf

- 是什么使您专注于网络技术的呢？

20 世纪 60 年代末，我在 UCLA 一直做程序员的工作。我的工作得到了美国国防部高级研究计划署（那时叫 ARPA，现在叫 DARPA）的支持。我那时在刚创建不久的 ARPAnet 的网络测量中心，为 Leonard Kleinrock 教授的实验室工作。ARPAnet 的第一个节点于 1969 年 9 月 1 日安装在 UCLA。我负责为计算机编程，以获取有关 ARPAnet 的性能信息，并报告这些信息以便与数学模型作比较，预测网络性能。

我和其他几名研究生负责研制所谓的 ARPAnet 主机级协议，该协议的过程和格式将使得网络中许多不同类型的计算机相互交互。这是我进入分布式计算与通信新世界中的一次迷人的探索。

- 当您第一次设计该协议时，您曾想象过 IP 会像今天这样变得无所不在吗？

当我和 Bob Kahn 于 1973 年最初从事该项工作时，我想我们的注意力大多集中在这样一个重要的问题上：假定我们不能实际改变这些网络本身，那么怎样才能让异构的分组网络彼此互操作呢？我们希望能找到一种方法可以使任意多的分组交换网以透明的方式进行互联，以便主机彼此之间不做任何转换就能进行端到端通信。我认为我们那时已经知道了我们正在处理强大的和可扩充的技术，但还没清楚地想过有数亿台计算机都连入因特网时的世界会是什么样。

- 您现在能预见网络与因特网的未来吗？您认为在它们的发展中存在的最大挑战或障碍是什么？

我相信因特网本身以及一般的网络都将要继续扩大。已有令人信服的证据表明，在因特网上将有数十亿个因特网使能设备，包括移动电话、冰箱、个人数字助理、家用服务器、电视等家用电器，以及大批通常的便携机、服务器等。重大挑战包括支持移动性、电池寿命、网络接入链路的容量、以不受限的方式扩展网络光学核心的能力。设计因特网的星际扩展是我在喷气推进实验室深入研究的一项计划。我们需要从 IPv4（32 比特地址）过渡到 IPv6（128 比特）。要做的事情实在是太多了！

- 是谁激发了您的职业灵感？

我的同事 Bob Kahn、我的论文导师 Gerald Estrin、我最好的朋友 Steve Crocker（我们在高中就认识了，1960 年是他带我进入了计算机学科之门！），以及数千名今天仍在继续推动因特网发展的工程师。

- 您对进入网络/因特网领域的学生有什么忠告吗？

要跳出现有系统的限制来思考问题，想一想什么是可行的；随后再做艰苦工作以谋划如何从事物的当前状态到达所想的状态。要敢于想象：我和喷气推进实验室的 6 个同事一直在从事陆地因特网的星际扩展设计。这也许要花几十年才能实现，任务会一个接着一个地出现，可以用这句话来总结：“一个人总是要不断地超越自我，否则还有什么乐趣可言？”

网络层：控制平面

在本章中，我们将通过包含网络层的**控制平面**组件来完成我们的网络层之旅。控制平面作为一种网络范围的逻辑，不仅控制沿着从源主机到目的主机的端到端路径间的路由器如何转发数据报，而且控制网络层组件和服务如何配置和管理。在5.2节中，我们将包含传统的计算图中最低开销路径的路由选择算法。这些算法是两个广为部署的因特网路由选择协议OSPF和BGP的基础，我们将分别在5.3节和5.4节中涉及。如我们将看到的那样，OSPF是一种运行在单一ISP的网络中的路由选择算法。BGP是一种在因特网中用于互联所有网络的路由选择算法，因此常被称为因特网的“黏合剂”。传统上，控制平面功能与数据平面的转发功能在一起实现，在路由器中作为统一的整体。如我们在第4章所学习的那样，软件定义网络（SDN）在数据平面和控制平面之间做了明确分割，在一台分离的“控制器”服务中实现了控制平面功能，该控制器服务与它所控制的路由器的转发组件完全分开并远离。我们将在5.5节中讨论SDN控制器。

在5.6节和5.7节中，我们将涉及管理IP网络的某些具体细节：ICMP（互联网控制报文协议）和SNMP（简单网络管理协议）。

5.1 概述

我们通过回顾图4-2和图4-3，迅速建立起学习网络控制平面的环境。在这里，我们看到了转发表（在基于目的地转发的场景中）和流表（在通用转发的场景中）是链接网络层的数据平面和控制平面的首要元素。我们知道这些表定义了一台路由器的本地数据平面转发行为。我们看到在通用转发的场景下，所采取的动作（4.4.2节）不仅包括转发一个分组到达路由器的每个输出端口，而且能够丢弃一个分组、复制一个分组和/或重写第2、3或4层分组首部字段。

在本章中，我们将学习这些转发表和流表是如何计算、维护和安装的。在4.1节的网络层概述中，我们已经学习了完成这些工作有两种可能的方法。

- *每路由器控制*。图5-1显示了在每台路由器中运行一种路由选择算法的情况，每台路由器中都包含转发和路由选择功能。每台路由器有一个路由选择组件，用于与其他路由器中的路由选择组件通信，以计算其转发表的值。这种每路由器控制的方法在因特网中已经使用了几十年。将在5.3节和5.4节中学习的OSPF和BGP协议都是基于这种每路由器的方法进行控制的。
- *逻辑集中式控制*。图5-2显示了逻辑集中式控制器计算并分发转发表以供每台路由器使用的情况。如我们在4.4节中所见，通用的“匹配加动作”抽象允许执行传统的IP转发以及其他功能（负载共享、防火墙功能和NAT）的丰富集合，而这些功能先前是在单独的中间盒中实现的。

该控制器经一种定义良好的协议与每台路由器中的一个控制代理（CA）进行交互，以配置和管理该路由器的转发表。CA一般具有最少的功能，其任务是与控制器通信并且

按控制器命令行事。与图 5-1 中的路由选择算法不同，这些 CA 既不能直接相互交互，也不能主动参与计算转发表。这是每路由器控制和逻辑集中式控制之间的关键差异。

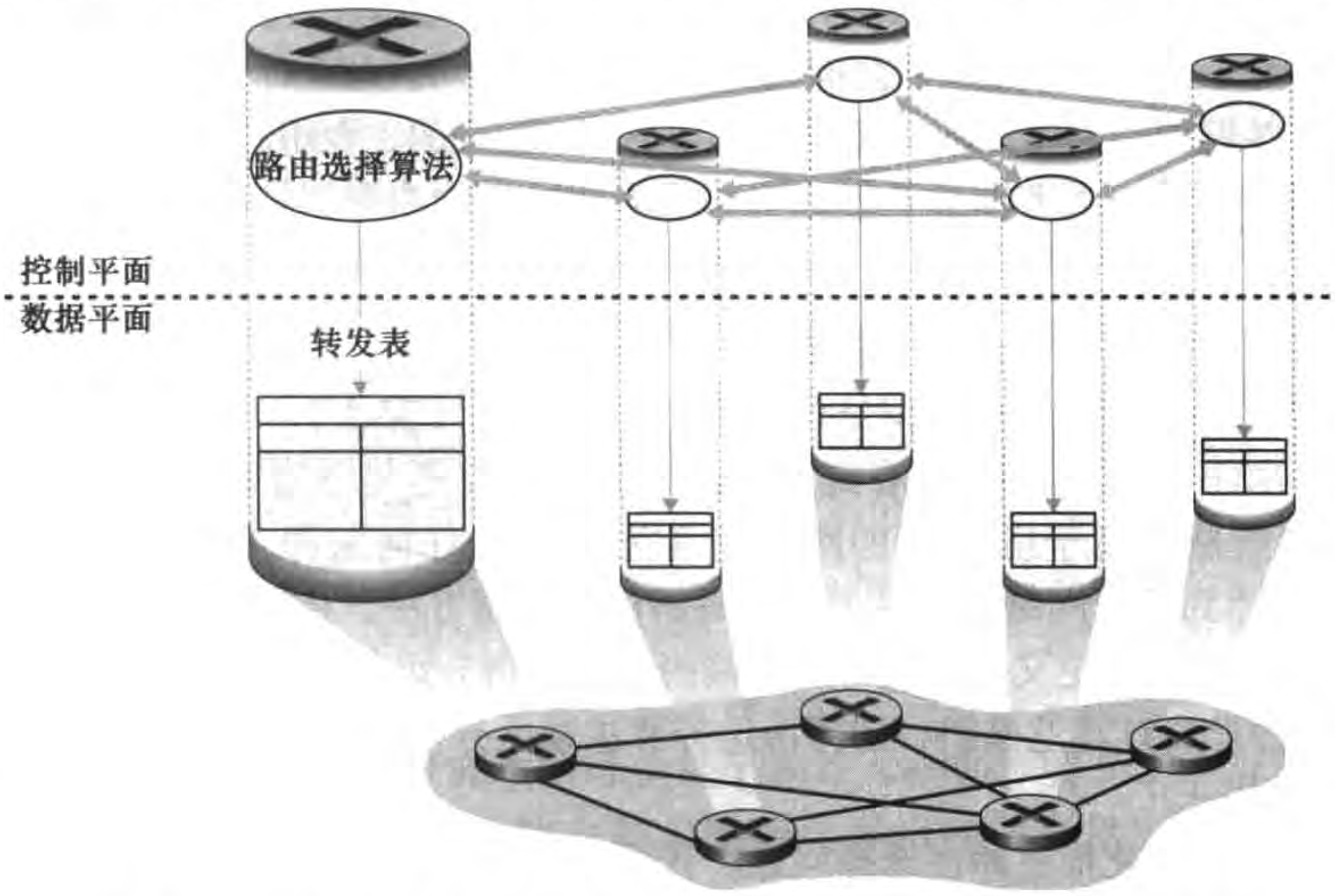


图 5-1 每路由器控制：在控制平面中各个路由选择算法组件相互作用

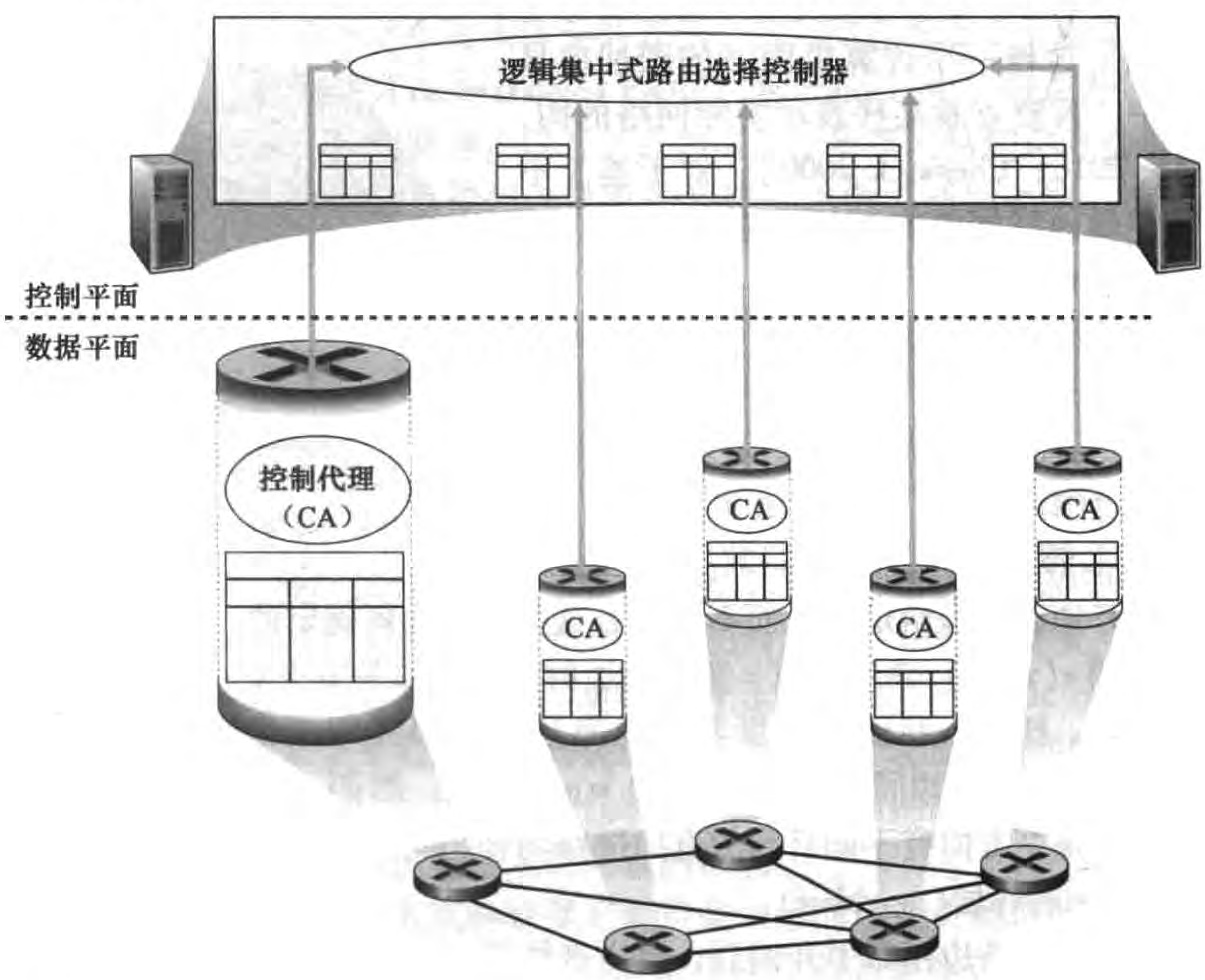


图 5-2 逻辑集中式控制：一个完全分开的（通常是远地的）控制器与本地控制代理交互

“逻辑集中式”控制［Levin 2012］意味着就像路由选择控制服务位于单一的集中服务点那样获取它们，即使该服务出于容错和性能扩展性的原因，很可能经由多个服务器实现。正如我们将在 5.5 节中所见，SDN 采用了逻辑集中式控制器的概念，而这种方法在生产部署中

得到了越来越多的应用。谷歌在它的内部 B4 全球广域网中使用了 SDN 控制路由器，该广域网互联了它的数据中心［Jain 2013］。来自微软研究院的 SWAN［Hong 2013］，使用了一个逻辑集中式控制器来管理广域网和数据中心网络之间的路由选择和转发。中国电信和中国联通在它们的数据中心内以及数据中心之间使用 SDN［Li 2015］。AT&T 已经通告［AT&T 2013］："支持许多 SDN 能力，支持 SDN 体系结构框架下独立定义的、专用的机制。"

5.2 路由选择算法

在本节中，我们将学习**路由选择算法**（routing algorithm），其目的是从发送方到接收方的过程中确定一条通过路由器网络的好的路径（等价于路由）。通常，一条好路径指具有最低开销的路径。然而我们将看到，实践中现实世界还关心诸如策略之类的问题（例如，有一个规则是"属于组织 Y 的路由器 X 不应转发任何来源于组织 Z 所属网络的分组"）。我们注意到无论网络控制平面采用每路由器控制方法，还是采用逻辑集中式控制方法，必定总是存在一条定义良好的一连串路由器，使得分组从发送主机到接收主机跨越网络"旅行"。因此，计算这些路径的路由选择算法是十分重要的，是最重要的 10 个十分重要的网络概念之一。

可以用图来形式化描述路由选择问题。我们知道**图**（graph）$G=(N, E)$ 是一个 N 个节点和 E 条边的集合，其中每条边是取自 N 的一对节点。在网络层路由选择的环境中，图中的节点表示路由器，这是做出分组转发决定的点；连接这些节点的边表示这些路由器之间的物理链路。这样一个计算机网络的图抽象显示在图 5-3 中。若要查看某些表示实际网络的图，参见［Dodge 2016；Cheswick 2000］；对于基于不同的图模型建模因特网的好坏的讨论，参见［Zegura 1997；Faloutsos 1999；Li 2004］。

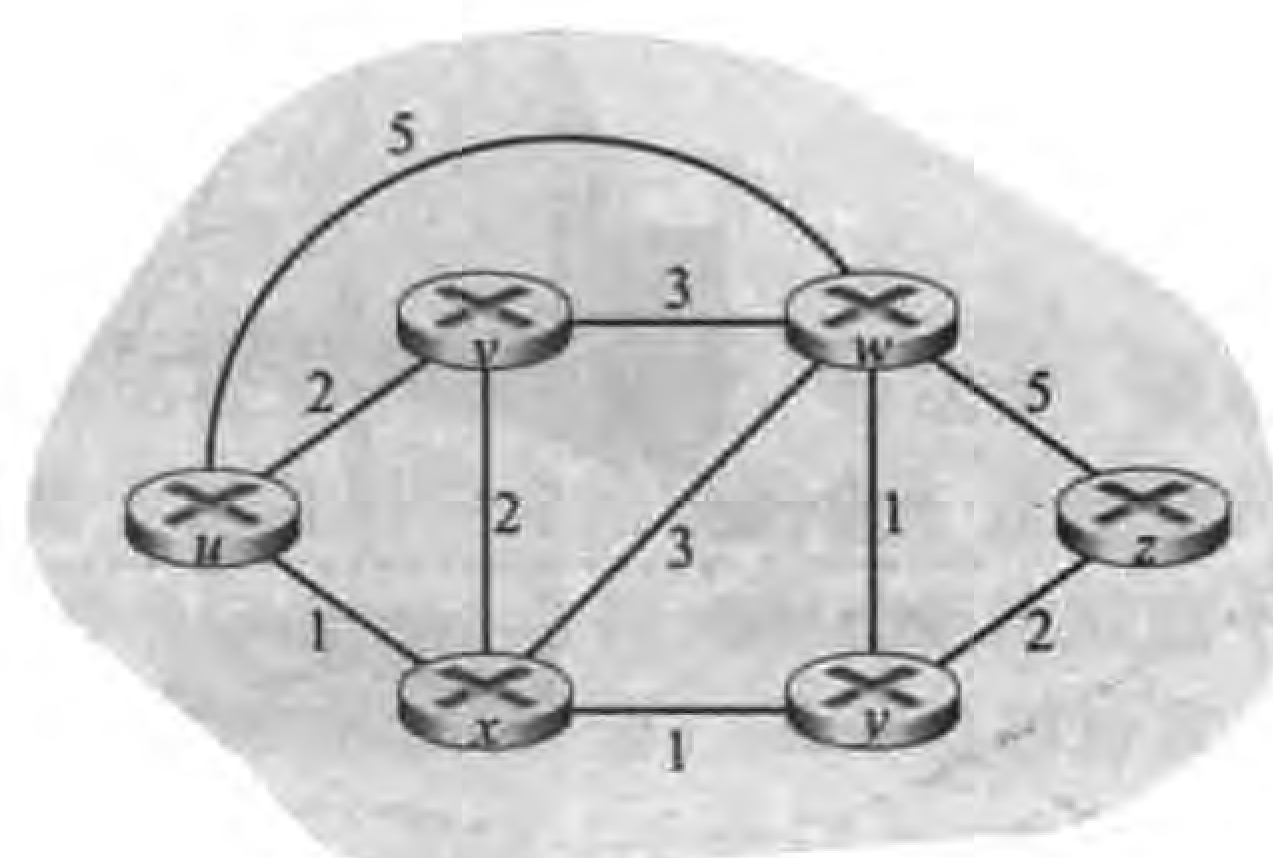


图 5-3　一个计算机网络的抽象图模型

如图 5-3 所示，一条边还有一个值表示它的开销。通常，一条边的开销可反映出对应链路的物理长度（例如一条越洋链路的开销可能比一条短途陆地链路的开销高），它的链路速度，或与该链路相关的金钱上的开销。为了我们的目的，我们只将这些链路开销看成是给定的，而不必操心这些值是如何确定的。对于 E 中的任一条边 (x, y)，我们用 $c(x, y)$ 表示节点 x 和 y 间边的开销。如果节点对 (x, y) 不属于 E，则置 $c(x, y)=\infty$。此外，我们在这里考虑的都是无向图（即图的边没有方向），因此边 (x, y) 与边 (y, x) 是相同的并且 $c(x, y)=c(y, x)$。然而，我们将学习的算法能够很容易地扩展到在每个方向有不同开销的有向链路场合。同时，如果 (x, y) 属于 E，节点 y 也被称为节点 x 的**邻居**（neighbor）。

在图抽象中为各条边指派了开销后，路由选择算法的天然目标是找出从源到目的地间的最低开销路径。为了使问题更为精确，回想在图 $G=(N, E)$ 中的一条**路径**（path）是一个节点序列 $(x_1, x_2, \cdots, x_p)$，这样每一个对 $(x_1, x_2), (x_2, x_3), \cdots, (x_{p-1}, x_p)$ 是 E 中的边。路径 $(x_1, x_2, \cdots, x_p)$ 的开销只是沿着路径所有边的开销的总和，即 $c(x_1, x_2)+c(x_2, x_3)+\cdots+c(x_{p-1}, x_p)$。给定任何两个节点 x 和 y，通常在这两个节点之间有许多条路径，每条路径都有一个开销。这些路径中的一条或多条是**最低开销路径**

(least-cost path)。因此最低开销路径问题是清楚的：找出源和目的地之间具有最低开销的一条路。例如，在图5-3中，源节点 u 和目的节点 w 之间的最低开销路径是（u，x，y，w），具有的路径开销是3。注意到若在图中的所有边具有相同的开销，则最低开销路径也就是**最短路径**（shortest path），即在源和目的地之间的具有最少链路数量的路径。

作为一个简单练习，试找出图5-3中从节点 u 到节点 z 的最低开销路径，并要反映出你是如何算出该路径的。如果你像大多数人一样，通过考察图5-3，跟踪几条从 u 到 z 的路由，你就能找出从 u 到 z 的路径，然后以某种方式来确信你所选择的路径就是所有可能的路径中具有最低开销的路径。(你考察过 u 到 z 之间的所有17条可能的路径吗？很可能没有!）这种计算就是一种集中式路由选择算法的例子，即路由选择算法在一个位置（你的大脑中）运行，该位置具有网络的完整信息。一般而言，路由选择算法的一种分类方式是根据该算法是集中式还是分散式来划分。

- **集中式路由选择算法**（centralized routing algorithm）用完整的、全局性的网络知识计算出从源到目的地之间的最低开销路径。也就是说，该算法以所有节点之间的连通性及所有链路的开销为输入。这就要求该算法在真正开始计算以前，要以某种方式获得这些信息。计算本身可在某个场点（例如，图5-2中所示的逻辑集中式控制器）进行，或在每台路由器的路由选择组件中重复进行（例如在图5-1中)。然而，这里的主要区别在于，集中式算法具有关于连通性和链路开销方面的完整信息。具有全局状态信息的算法常被称作**链路状态**（Link State，LS）**算法**，因为该算法必须知道网络中每条链路的开销。我们将在5.2.1节中学习LS算法。
- 在**分散式路由选择算法**（decentralized routing algorithm）中，路由器以迭代、分布式的方式计算出最低开销路径。没有节点拥有关于所有网络链路开销的完整信息。相反，每个节点仅有与其直接相连链路的开销知识即可开始工作。然后，通过迭代计算过程以及与相邻节点的信息交换，一个节点逐渐计算出到达某目的节点或一组目的节点的最低开销路径。我们将在后面的5.2.2节学习一个称为**距离向量**(Distance-Vector，DV）**算法**的分散式路由选择算法。之所以叫作DV算法，是因为每个节点维护到网络中所有其他节点的开销（距离）估计的向量。这种分散式算法，通过相邻路由器之间的交互式报文交换，也许更为天然地适合那些路由器直接交互的控制平面，就像在图5-1中那样。

路由选择算法的第二种广义分类方式是根据算法是静态的还是动态的进行分类。在**静态路由选择算法**（static routing algorithm）中，路由随时间的变化非常缓慢，通常是人工进行调整（如人为手工编辑一条链路开销)。**动态路由选择算法**（dynamic routing algorithm）随着网络流量负载或拓扑发生变化而改变路由选择路径。一个动态算法可周期性地运行或直接响应拓扑或链路开销的变化而运行。虽然动态算法易于对网络的变化做出反应，但也更容易受诸如路由选择循环、路由振荡之类问题的影响。

路由选择算法的第三种分类方式是根据它是负载敏感的还是负载迟钝的进行划分。在**负载敏感算法**（load-sensitive algorithm）中，链路开销会动态地变化以反映出底层链路的当前拥塞水平。如果当前拥塞的一条链路与高开销相联系，则路由选择算法趋向于绕开该拥塞链路来选择路由。而早期的ARPAnet路由选择算法就是负载敏感的［McQuillan 1980]，所以遇到了许多难题［Huitema 1998］。当今的因特网路由选择算法（如RIP、OSPF和BGP）都是**负载迟钝的**（load-insensitive)，因为某条链路的开销不明确地反映其当前（或最近）的拥塞水平。

5.2.1 链路状态路由选择算法

前面讲过，在链路状态算法中，网络拓扑和所有的链路开销都是已知的，也就是说可用作LS算法的输入。实践中这是通过让每个节点向网络中所有其他节点广播链路状态分组来完成的，其中每个链路状态分组包含它所连接的链路的标识和开销。在实践中（例如使用因特网的OSPF路由选择协议，讨论见5.3节），这经常由**链路状态广播**（link state broadcast）算法［Perlman 1999］来完成。节点广播的结果是所有节点都具有该网络的统一、完整的视图。于是每个节点都能够像其他节点一样，运行LS算法并计算出相同的最低开销路径集合。

我们下面给出的链路状态路由选择算法叫作Dijkstra算法，该算法以其发明者命名。一个密切相关的算法是Prim算法，有关图算法的一般性讨论参见［Cormen 2001］。Dijkstra算法计算从某节点（源节点，我们称之为u）到网络中所有其他节点的最低开销路径。Dijkstra算法是迭代算法，其性质是经算法的第k次迭代后，可知道到k个目的节点的最低开销路径，在到所有目的节点的最低开销路径之中，这k条路径具有k个最低开销。我们定义下列记号。

- $D(v)$：到算法的本次迭代，从源节点到目的节点v的最低开销路径的开销。
- $p(v)$：从源到v沿着当前最低开销路径的前一节点（v的邻居）。
- N'：节点子集；如果从源到v的最低开销路径已确知，v在N'中。

该集中式路由选择算法由一个初始化步骤和其后的循环组成。循环执行的次数与网络中节点个数相同。一旦终止，该算法就计算出了从源节点u到网络中每个其他节点的最短路径。

源节点u的链路状态（LS）算法

```
Initialization:
 N' = {u}
 for all nodes v
   if v is a neighbor of u
     then D(v) = c(u,v)
   else D(v) = ∞

Loop
 find w not in N' such that D(w) is a minimum
 add w to N'
 update D(v) for each neighbor v of w and not in N':
       D(v) = min(D(v), D(w)+ c(w,v) )
  /* new cost to v is either old cost to v or known
   least path cost to w plus cost from w to v */
until N'= N
```

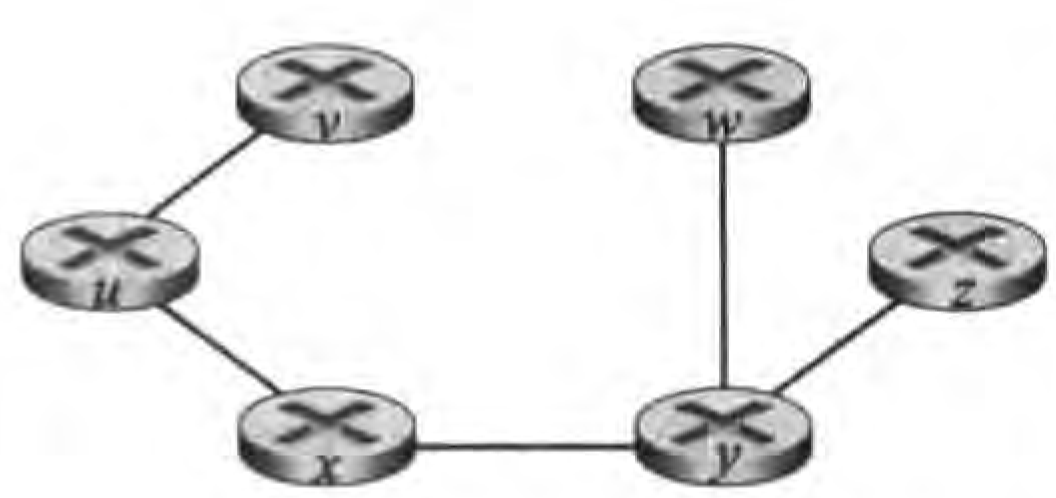
举一个例子，考虑图5-3中的网络，计算从u到所有可能目的地的最低开销路径。该算法的计算过程以表格方式总结于表5-1中，表中的每一行给出了迭代结束时该算法的变量的值。我们详细地考虑前几个步骤。

表5-1 在图5-3中的网络上运行的链路状态算法

步骤	N'	$D(v)$，$p(v)$	$D(w)$，$p(w)$	$D(x)$，$p(x)$	$D(y)$，$p(y)$	$D(z)$，$p(z)$
0	u	2，u	5，u	1，u	∞	∞
1	ux	2，u	4，x		2，x	∞
2	uxy	2，u	3，y			4，y
3	$uxyv$		3，y			4，y
4	$uxyvw$					4，y
5	$uxyvwz$					

- 在初始化步骤，从 u 到与其直接相连的邻居 v、x、w 的当前已知最低开销路径分别初始化为2、1和5。特别值得注意的是，到 w 的开销被设为5（尽管我们很快就会看见确实存在一条开销更小的路径），因为这是从 u 到 w 的直接（一跳）链路开销。到 y 与 z 的开销被设为无穷大，因为它们不直接与 u 连接。
- 在第一次迭代中，我们观察那些还未加到集合 N' 中的节点，并且找出在前一次迭代结束时具有最低开销的节点。那个节点便是 x，其开销是1，因此 x 被加到集合 N' 中。于是 LS 算法中的第12行中的程序被执行，以更新所有节点 v 的 $D(v)$，产生表5-1中第2行（步骤1）所示的结果。到 v 的路径开销未变。经过节点 x 到 w（在初始化结束时其开销为5）的路径开销被发现为4。因此这条具有更低开销的路径被选中，且沿从 u 开始的最短路径上 w' 的前一节点被设为 x。类似地，到 y（经过 x）的开销被计算为2，且该表也被相应地更新。
- 在第二次迭代时，节点 v 与 y 被发现具有最低开销路径（2），并且我们任意改变次序将 y 加到集合 N' 中，使得 N' 中含有 u、x 和 y。到仍不在 N' 中的其余节点（即节点 v、w 和 z）的开销通过 LS 算法中的第12行进行更新，产生如表5-1中第3行所示的结果。
- 如此等等。

当 LS 算法终止时，对于每个节点，我们都得到从源节点沿着它的最低开销路径的前一节点。对于每个前一节点，我们又有它的前一节点，以此方式我们可以构建从源节点到所有目的节点的完整路径。通过对每个目的节点存放从 u 到目的地的最低开销路径上的下一跳节点，在一个节点（如节点 u）中的转发表则能够根据此信息而构建。图5-4显示了对于图5-3中的网络产生的最低开销路径和 u 中的转发表。



目的地	链路
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

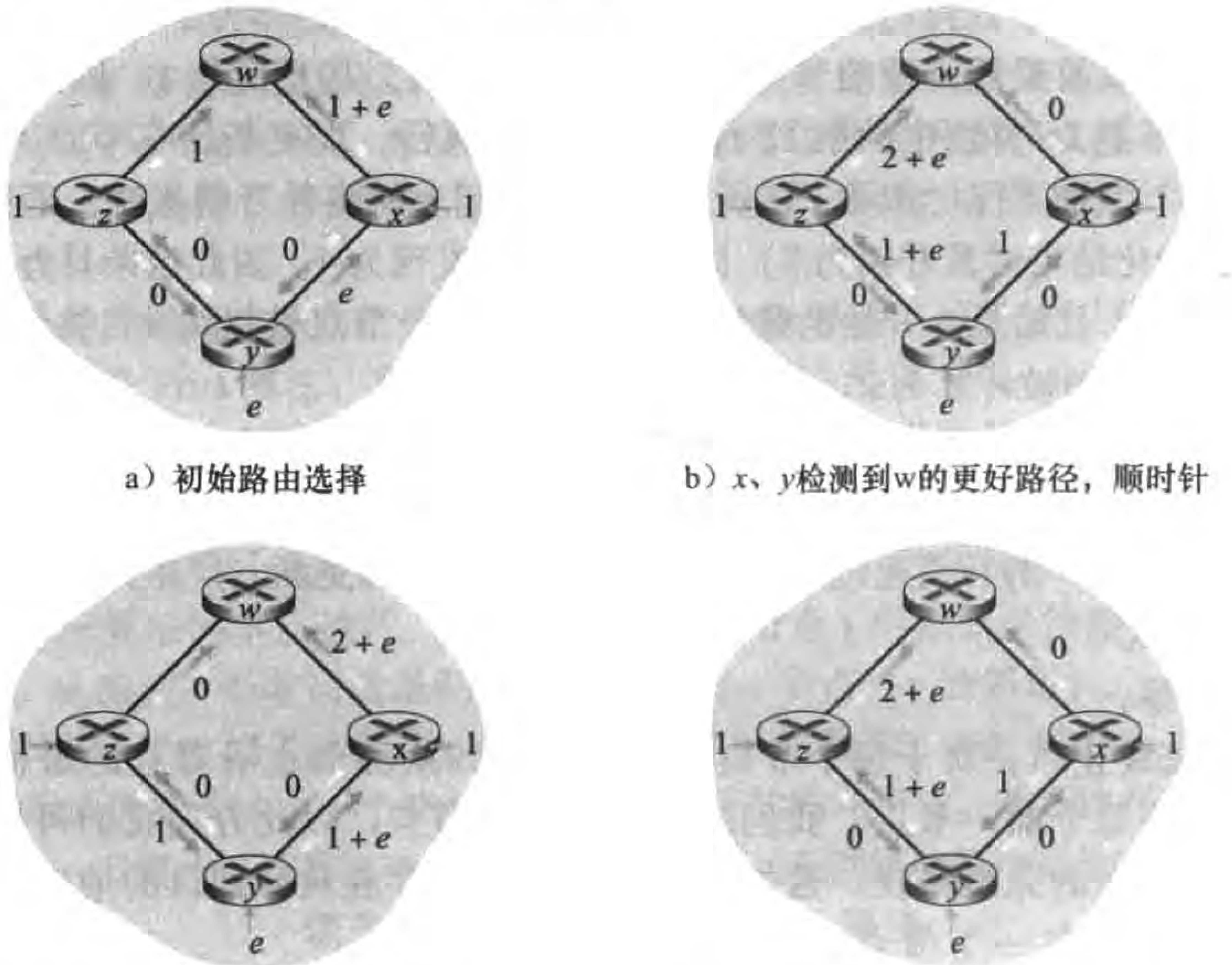
图5-4 对于节点 u 的最低开销路径和转发表

该算法的计算复杂性是什么？即给定 n 个节点（不算源节点），在最坏情况下要经过多少次计算，才能找到从源节点到所有目的节点的最低开销路径？在第一次迭代中，我们需要搜索所有的 n 个节点以确定出不在 N' 中且具有最低开销的节点 w。在第二次迭代时，我们需要检查 $n-1$ 个节点以确定最低开销。第三次对 $n-2$ 个节点迭代，依次类推。总之，我们在所有迭代中需要搜寻的节点总数为 $n(n+1)/2$，因此我们说前面实现的链路状态算法在最差情况下复杂性为 $O(n^2)$。（该算法的一种更复杂的实现是使用一种称为堆的数据结构，能用对数时间而不是线性时间得到第9行的最小值，因而减少其复杂性。）

在完成 LS 算法的讨论之前，我们考虑一下可能出现的问题。图5-5显示了一个简单的网络拓扑，图中的链路开销等于链路上承载的负载，例如反映要历经的时延。在该例中，链路开销是非对称的，即仅当在链路（u，v）两个方向所承载的负载相同时 $c(u, v)$ 与 $c(v, u)$ 才相等。在该例中，节点 z 产生发往 w 的一个单元的流量，节点 x 也产生发往 w 的一个单元的流量，并且节点 y 也产生发往 w 的一个数量为 e 的流量。初始路由选择情况如图5-5a所示，其链路开销对应于承载的流量。

当 LS 算法再次运行时，节点 y 确定（基于图5-5a所示的链路开销）顺时针到 w 的路径开销为1，而逆时针到 w 的路径开销（一直使用的）是 $1+e$。因此 y 到 w 的最低开销路

径现在是顺时针的。类似地，x 确定其到 w 的新的最低开销路径也是顺时针的，产生如图 5-5b 中所示的开销。当 LS 算法下次运行时，节点 x、y 和 z 都检测到一条至 w 的逆时针方向零开销路径，它们都将其流量引导到逆时针方向的路由上。下次 LS 算法运行时，x、y 和 z 都将其流量引导到顺时针方向的路由上。



a）初始路由选择

b）x、y检测到w的更好路径，顺时针

c）x、y、z检测到w的更好路径，逆时针

d）x、y、z检测到w的更好路径，顺时针

图 5-5　拥塞敏感的路由选择的振荡

如何才能防止这样的振荡（它不只是出现在链路状态算法中，而且也可能出现在任何使用拥塞或基于时延的链路测度的算法中）。一种解决方案可能强制链路开销不依赖于所承载的流量，但那是一种不可接受的解决方案，因为路由选择的目标之一就是要避开高度拥塞（如高时延）的链路。另一种解决方案就是确保并非所有的路由器都同时运行 LS 算法。这似乎是一个更合理的方案，因为我们希望即使路由器以相同周期运行 LS 算法，在每个节点上算法执行的时机也将是不同的。有趣的是，研究人员近来已注意到了因特网上的路由器能在它们之间进行自同步［Floyd Synchronization 1994］。这就是说，即使它们初始时以同一周期但在不同时刻执行算法，算法执行时机最终会在路由器上变为同步并保持之。避免这种自同步的一种方法是，让每台路由器发送链路通告的时间随机化。

学习过 LS 算法之后，我们接下来考虑目前在实践中使用的其他重要的路由选择算法，即距离向量路由选择算法。

5.2.2　距离向量路由选择算法

距离向量（Distance-Vector，DV）算法是一种迭代的、异步的和分布式的算法，而 LS 算法是一种使用全局信息的算法。说它是分布式的，是因为每个节点都要从一个或多个直接相连邻居接收某些信息，执行计算，然后将其计算结果分发给邻居。说它是迭代的，是

因为此过程一直要持续到邻居之间无更多信息要交换为止。（有趣的是，此算法是自我终止的，即没有计算应该停止的信号，它就停止了。）说它是异步的，是因为它不要求所有节点相互之间步伐一致地操作。我们将看到一个异步的、迭代的、自我终止的、分布式的算法比一个集中式的算法要有趣得多！

在我们给出 DV 算法之前，有必要讨论一下存在于最低开销路径的开销之间的一种重要关系。令 $d_x(y)$ 是从节点 x 到节点 y 的最低开销路径的开销。则该最低开销与著名的 Bellman-Ford 方程相关，即

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\} \tag{5-1}$$

方程中的 $\min_v$ 是对于 x 的所有邻居的。Bellman-Ford 方程是相当直观的。实际上，从 x 到 v 遍历之后，如果我们接下来取从 v 到 y 的最低开销路径，则该路径开销将是 $c(x, v) + d_v(y)$。因此我们必须通过遍历某些邻居 v 开始，从 x 到 y 的最低开销是对所有邻居 v 的 $c(x, v) + d_v(y)$ 的最小值。

但是对于那些可能怀疑该方程正确性的人，我们核查在图 5-3 中的源节点 u 和目的节点 z。源节点 u 有 3 个邻居：节点 v、x 和 w。通过遍历该图中的各条路径，容易看出 $d_v(z) = 5$、$d_x(z) = 3$ 和 $d_v(z) = 3$。将这些值连同开销 $c(u, v) = 2$、$c(u, x) = 1$ 和 $c(u, w) = 5$ 代入方程（5-1），得出 $d_u(z) = \min\{2+5, 5+3, 1+3\} = 4$，这显然是正确的，并且对同一个网络来说，这正是 Dijkstra 算法为我们提供的结果。这种快速验证应当有助于消除你可能具有的任何怀疑。

Bellman-Ford 方程不止是一种智力上的珍品，它实际上具有重大的实践重要性。特别是对 Bellman-Ford 方程的解为节点 x 的转发表提供了表项。为了理解这一点，令 v^* 是取得方程（5-1）中最小值的任何相邻节点。接下来，如果节点 x 要沿着最低开销路径向节点 y 发送一个分组，它应当首先向节点 v^* 转发该分组。因此，节点 x 的转发表将指定节点 v^* 作为最终目的地 y 的下一跳路由器。Bellman-Ford 方程的另一个重要实际贡献是，它提出了将在 DV 算法中发生的邻居到邻居通信的形式。

其基本思想如下。每个节点 x 以 $D_x(y)$ 开始，对在 N 中的所有节点 y，估计从 x 到 y 的最低开销路径的开销。令 $\boldsymbol{D}_x = [D_x(y): y \in N]$ 是节点 x 的距离向量，该向量是从 x 到在 N 中的所有其他节点 y 的开销估计向量。使用 DV 算法，每个节点 x 维护下列路由选择信息：

- 对于每个邻居 v，从 x 到直接相连邻居 v 的开销为 $c(x, v)$。
- 节点 x 的距离向量，即 $\boldsymbol{D}_x = [D_x(y): y \in N]$，包含了 x 到 N 中所有目的地 y 的开销估计值。
- 它的每个邻居的距离向量，即对 x 的每个邻居 v，有 $\boldsymbol{D}_v = [D_v(y): y \in N]$。

在该分布式、异步算法中，每个节点不时地向它的每个邻居发送它的距离向量副本。当节点 x 从它的任何一个邻居 v 接收到一个新距离向量，它保存 v 的距离向量，然后使用 Bellman-Ford 方程更新它自己的距离向量如下：

$$D_x(y) = \min_v \{c(x,v) + D_v(y)\} \quad 对 N 中的每个节点$$

如果节点 x 的距离向量因这个更新步骤而改变，节点 x 接下来将向它的每个邻居发送其更新后的距离向量，这继而让所有邻居更新它们自己的距离向量。令人惊奇的是，只要所有的节点继续以异步方式交换它们的距离向量，每个开销估计 $D_x(y)$ 收敛到 $d_x(y)$，$d_x(y)$ 为从节点 x 到节点 y 的实际最低开销路径的开销 [Bersekas 1991]！

距离向量（DV）算法

在每个节点 x：

```
1  Initialization:
2    for all destinations y in N:
3       Dx(y)= c(x,y)/* if y is not a neighbor then c(x,y)= ∞ */
4    for each neighbor w
5       Dw(y) = ? for all destinations y in N
6    for each neighbor w
7       send distance vector  Dx = [Dx(y): y in N] to w
8
9  loop
10    wait  (until I see a link cost change to some neighbor w or
11            until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14        Dx(y) = minv{c(x,v) + Dv(y)}
15
16 if Dx(y) changed for any destination y
17       send distance vector Dx  = [Dx(y): y in N] to all neighbors
18
19 forever
```

在该 DV 算法中，当节点 x 发现它的直接相连的链路开销变化或从某个邻居接收到一个距离向量的更新时，它就更新其距离向量估计值。但是为了一个给定的目的地 y 而更新它的转发表，节点 x 真正需要知道的不是到 y 的最短路径距离，而是沿着最短路径到 y 的下一跳路由器邻居节点 $v^*(y)$。如你可能期望的那样，下一跳路由器 $v^*(y)$ 是在 DV 算法第 14 行中取得最小值的邻居 v。（如果有多个取得最小值的邻居 v，则 $v^*(y)$ 能够是其中任何一个有最小值的邻居。）因此，对于每个目的地 y，在第 13～14 行中，节点 x 也决定 $v^*(y)$ 并更新它对目的地 y 的转发表。

前面讲过 LS 算法是一种全局算法，在于它要求每个节点在运行 Dijkstra 算法之前，首先获得该网络的完整信息。DV 算法是*分布式的*，它不使用这样的全局信息。实际上，节点具有的唯一信息是它到直接相连邻居的链路开销和它从这些邻居接收到的信息。每个节点等待来自任何邻居的更新（第 10～11 行），当接收到一个更新时计算它的新距离向量（第 14 行）并向它的邻居分布其新距离向量（第 16～17 行）。在实践中许多类似 DV 的算法被用于多种路由选择协议中，包括因特网的 RIP 和 BGP、ISO IDRP、Novell IPX 和早期的 ARPAnet。

图 5-6 举例说明了 DV 算法的运行，应用场合是该图顶部有三个节点的简单网络。算法的运行以同步的方式显示出来，其中所有节点同时从其邻居接收报文，计算其新距离向量，如果距离向量发生了变化则通知其邻居。学习完这个例子后，你应当确信该算法以异步方式也能正确运行，异步方式中可在任意时刻出现节点计算与更新的产生/接收。

该图最左边一列显示了这 3 个节点各自的初始**路由选择表**（routing table）。例如，位于左上角的表是节点 x 的初始路由选择表。在一张特定的路由选择表中，每行是一个距离向量——特别是每个节点的路由选择表包括了它的距离向量和它的每个邻居的距离向量。因此，在节点 x 的初始路由选择表中的第一行是 $\boldsymbol{D}_x=[D_x(x), D_x(y), D_x(z)]=[0, 2, 7]$。在该表的第二和第三行是最近分别从节点 y 和 z 收到的距离向量。因为在初始化时节点 x 还没有从节点 y 和 z 收到任何东西，所以第二行和第三行表项中被初始化为无穷大。

初始化后，每个节点向它的两个邻居发送其距离向量。图 5-6 中用从表的第一列到表的第二列的箭头说明了这一情况。例如，节点 x 向两个节点 y 和 z 发送了它的距离向量

$D_x=[0, 2, 7]$。在接收到该更新后，每个节点重新计算它自己的距离向量。例如，节点 x 计算

$$D_x(x)=0$$

$$D_x(y)=\min\{c(x,y)+D_y(y), c(x,z)+D_z(y)\}=\min\{2+0, 7+1\}=2$$

$$D_x(z)=\min\{c(x,y)+D_y(z), c(x,z)+D_z(z)\}=\min\{2+1, 7+0\}=3$$

第二列因此为每个节点显示了节点的新距离向量连同刚从它的邻居接收到的距离向量。注意到，例如节点 x 到节点 z 的最低开销估计 $D_x(z)$ 已经从 7 变成了 3。还应注意到，对于节点 x，节点 y 在该 DV 算法的第 14 行中取得了最小值；因此在该算法的这个阶段，我们在节点 x 得到了 $v^*(y)=y$ 和 $v^*(z)=y$。

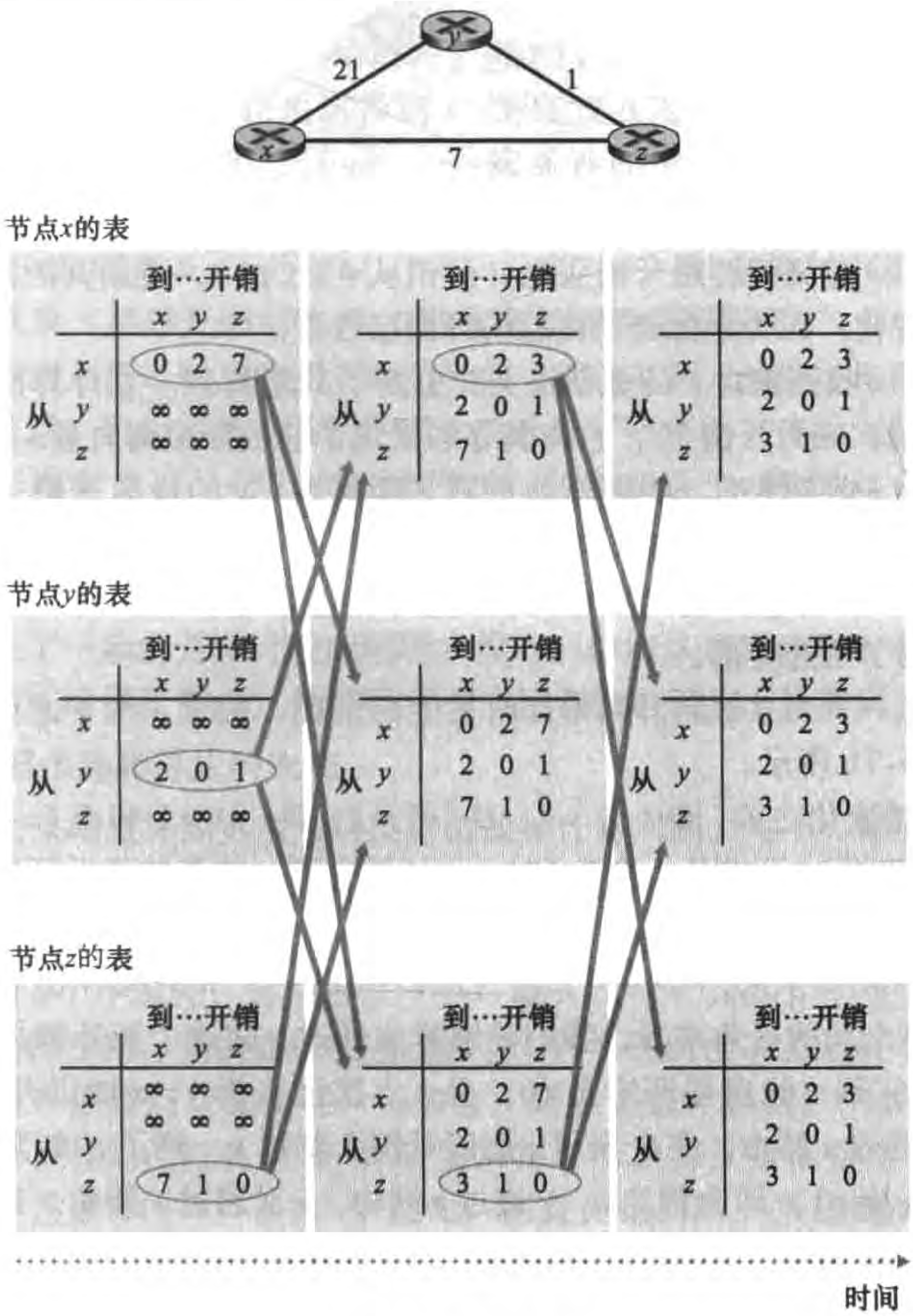


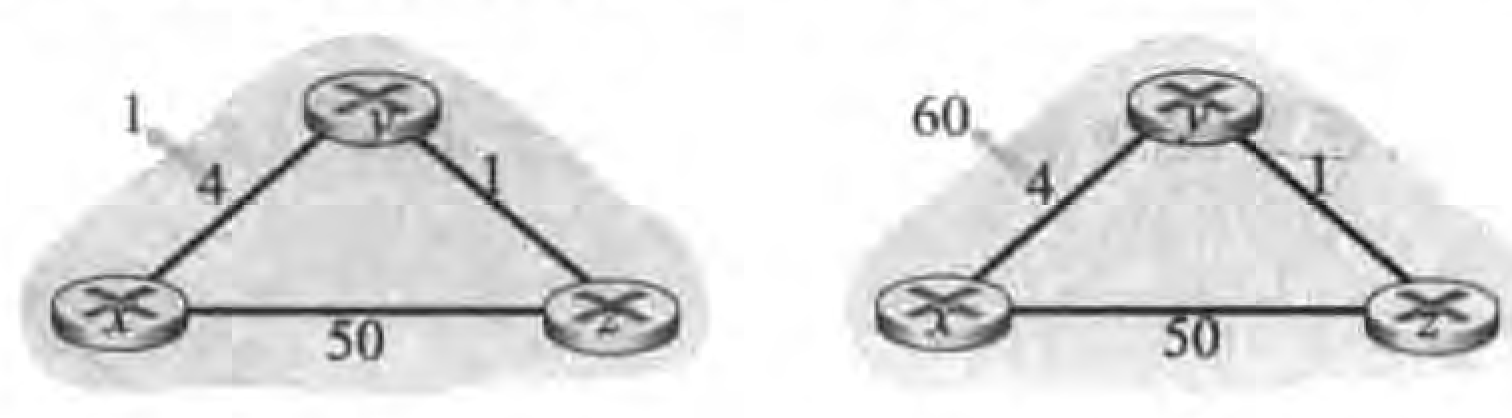
图 5-6　DV 算法

在节点重新计算它们的距离向量之后，它们再次向其邻居发送它们的更新距离向量（如果它们已经改变的话）。图 5-6 中由从表第二列到表第三列的箭头说明了这一情况。注意到仅有节点 x 和节点 z 发送了更新：节点 y 的距离向量没有发生变化，因此节点 y 没有发送更新。在接收到这些更新后，这些节点则重新计算它们的距离向量并更新它们的路由选择表，这些显示在第三列中。

从邻居接收更新距离向量、重新计算路由选择表项和通知邻居到目的地的最低开销路径的开销已经变化的过程继续下去，直到无更新报文发送为止。在这个时候，因为无更新报文发送，将不会出现进一步的路由选择表计算，该算法将进入静止状态，即所有的节点将执行DV算法的第10~11行中的等待。该算法停留在静止状态，直到一条链路开销发生改变，如下面所讨论的那样。

1. 距离向量算法：链路开销改变与链路故障

当一个运行DV算法的节点检测到从它自己到邻居的链路开销发生变化时（第10~11行），它就更新其距离向量（第13~14行），并且如果最低开销路径的开销发生了变化，向邻居通知其新的距离向量（第16~17行）。图5-7a图示了从y到x的链路开销从4变为1的情况。我们在此只关注y与z到目的地x的距离表中的有关表项。该DV算法导致下列事件序列的出现：



a)　　b)

图5-7　链路开销改变

- 在t_0时刻，y检测到链路开销变化（开销从4变为1），更新其距离向量，并通知其邻居这个变化，因为最低开销路径的开销已改变。
- 在t_1时刻，z收到来自y的更新报文并更新了其距离表。它计算出到x的新最低开销（从开销5减为开销2），它向其邻居发送了它的新距离向量。
- 在t_2时刻，y收到来自z的更新并更新其距离表。y的最低开销未变，因此y不发送任何报文给z。该算法进入静止状态。

因此，对于该DV算法只需两次迭代就到达了静止状态。在x与y之间开销减少的好消息通过网络得到了迅速传播。

我们现在考虑一下当某链路开销增加时发生的情况。假设x与y之间的链路开销从4增加到60，如图5-7b所示。

1）在链路开销变化之前，$D_y(x)=4$，$D_y(z)=1$，$D_z(y)=1$和$D_z(x)=5$。在t_0时刻，y检测到链路开销变化（开销从4变为60）。y计算它到x的新的最低开销路径的开销，其值为

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{60+0, 1+5\} = 6$$

当然，从网络全局的视角来看，我们能够看出经过z的这个新开销是错误的。但节点y仅有的信息是：它到x的直接开销是60，且z上次已告诉y，z能以开销5到x。因此，为了到达x，y将通过z路由，完全期望z能以开销5到达x。到了t_1时刻，我们遇到**路由选择环路**（routing loop），即为到达x，y通过z路由，z又通过y路由。路由选择环路就像一个黑洞，即目的地为x的分组在t_1时刻到达y或z后，将在这两个节点之间不停地（或直到转发表发生改变为止）来回反复。

2）因为节点y已算出到x的新的最低开销，它在t_1时刻将该新距离向量通知z。

3）在t_1后某个时间，z收到y的新距离向量，它指示了y到x的最低开销是6。z知道它能以开销1到达y，因此计算出到x的新最低开销$D_z(x)=\min\{50+0, 1+6\}=7$。因为$z$到$x$的最低开销已增加了，于是它便在$t_2$时刻通知$y$其新开销。

4）以类似方式，在收到z的新距离向量后，y决定$D_y(x)=8$并向z发送其距离向量。接下来z确定$D_z(x)=9$并向y发送其距离向量，等等。

该过程将要继续多久呢？你应认识到该循环将持续44次迭代（在 y 与 z 之间交换报文），即直到 z 最终算出它经由 y 的路径开销大于50为止。此时，z 将（最终）确定它到 x 的最低开销路径是经过它到 x 的直接连接。y 将经由 z 路由选择到 x。关于链路开销增加的坏消息的确传播得很慢！如果链路开销 $c(y, x)$ 从4变为10 000且开销 $c(z, x)$ 为9999时将发生什么样的现象呢？由于这种情况，我们所见的问题有时被称为无穷计数（count-to-infinity）问题。

2. 距离向量算法：增加毒性逆转

刚才描述的特定循环的场景可以通过使用一种称为*毒性逆转*（poisoned reverse）的技术而加以避免。其思想较为简单：如果 z 通过 y 路由选择到目的地 x，则 z 将通告 y，它（即 z）到 x 的距离是无穷大，也就是 z 将向 y 通告 $D_z(x)=\infty$（即使 z 实际上知道 $D_z(x)=5$）。只要 z 经 y 路由选择到 x，z 就持续地向 y 讲述这个善意的小谎言。因为 y 相信 z 没有到 x 的路径，故只要 z 继续经 y 路由选择到 x（并这样去撒谎），y 将永远不会试图经由 z 路由选择到 x。

我们现在看一下毒性逆转如何解决我们前面在图5-7b中遇到的特定环路问题。作为毒性逆转的结果，y 的距离表指示了 $D_z(x)=\infty$。当 (x, y) 链路的开销在 t_0 时刻从4变为60时，y 更新其表，虽然开销高达60，仍继续直接路由选择到 x，并将到 x 的新开销通知 z，即 $D_y(x)=60$。z 在 t_1 时刻收到更新后，便立即将其到 x 的路由切换到经过开销为50的直接 (z, x) 链路。因为这是一条新的到 x 的最低开销路径，且因为路径不再经过 y，z 就在 t_2 时刻通知 y 现在 $D_z(x)=50$。在收到来自 z 的更新后，y 使用 $D_y(x)=51$ 更新其距离表。另外，因为 z 此时位于 y 到 x 的最低开销路径上，所以 y 通过在 t_3 时刻通知 z 其 $D_y(x)=\infty$（即使 y 实际上知道 $D_y(x)=51$）毒化从 z 到 x 的逆向路径。

毒性逆转解决了一般的无穷计数问题吗？没有。你应认识到涉及3个或更多节点（而不只是两个直接相连的邻居节点）的环路将无法用毒性逆转技术检测到。

3. LS与DV路由选择算法的比较

DV和LS算法采用互补的方法来解决路由选择计算问题。在DV算法中，每个节点仅与它的直接相连的邻居交谈，但它为其邻居提供了从它自己到网络中（它所知道的）所有其他节点的最低开销估计。LS算法需要全局信息。因此，当在每台路由器中实现时，例如像在图4-2和图5-1中那样，每个节点（经广播）与所有其他节点通信，但仅告诉它们与它直接相连链路的开销。我们通过快速比较它们各自的属性来总结所学的链路状态与距离向量算法。记住 N 是节点（路由器）的集合，而 E 是边（链路）的集合。

- *报文复杂性*。我们已经看到LS算法要求每个节点都知道网络中每条链路的开销。这就要求要发送 $O(|N||E|)$ 个报文。而且无论何时一条链路的开销改变时，必须向所有节点发送新的链路开销。DV算法要求在每次迭代时，在两个直接相连邻居之间交换报文。我们已经看到，算法收敛所需时间依赖于许多因素。当链路开销改变时，DV算法仅当在新的链路开销导致与该链路相连节点的最低开销路径发生改变时，才传播已改变的链路开销。
- *收敛速度*。我们已经看到LS算法的实现是一个要求 $O(|N||E|)$ 个报文的 $O(|N|^2)$ 算法。DV算法收敛较慢，且在收敛时会遇到路由选择环路。DV算法还会遭遇无穷计数的问题。
- *健壮性*。如果一台路由器发生故障、行为错乱或受到蓄意破坏时情况会怎样呢？对

于LS算法，路由器能够向其连接的链路（而不是其他链路）广播不正确的开销。作为LS广播的一部分，一个节点也可损坏或丢弃它收到的任何LS广播分组。但是一个LS节点仅计算自己的转发表；其他节点也自行执行类似的计算。这就意味着在LS算法下，路由计算在某种程度上是分离的，提供了一定程度的健壮性。在DV算法下，一个节点可向任意或所有目的节点通告其不正确的最低开销路径。（在1997年，一个小ISP的一台有故障的路由器的确向美国的主干路由器提供了错误的路由选择信息。这引起了其他路由器将大量流量引向该故障路由器，并导致因特网的大部分中断连接达数小时［Neumann 1997］。）更一般地，我们会注意到每次迭代时，在DV算法中一个节点的计算会传递给它的邻居，然后在下次迭代时再间接地传递给邻居的邻居。在此情况下，DV算法中一个不正确的节点计算值会扩散到整个网络。

总之，两个算法没有一个是明显的赢家，它们的确都在因特网中得到了应用。

5.3　因特网中自治系统内部的路由选择：OSPF

在我们至今为止的算法研究中，我们将网络只看作一个互联路由器的集合。从所有路由器执行相同的路由选择算法以计算穿越整个网络的路由选择路径的意义上来说，一台路由器很难同另一台路由器区别开来。在实践中，该模型和这种一组执行同样路由选择算法的同质路由器集合的观点有一点简单化，有以下两个重要原因：

- **规模**。随着路由器数目变得很大，涉及路由选择信息的通信、计算和存储的开销将高得不可实现。当今的因特网由数亿台主机组成。在这些主机中存储的路由选择信息显然需要巨大容量的内存。在所有路由器之间广播连通性和链路开销更新所要求的负担将是巨大的！在如此大量的路由器中迭代的距离向量算法将肯定永远无法收敛！显然，必须采取一些措施以减少像因特网这种大型网络中的路由计算的复杂性。
- **管理自治**。如在1.3节描述的那样，因特网是ISP的网络，其中每个ISP都有它自己的路由器网络。ISP通常希望按自己的意愿运行路由器（如在自己的网络中运行它所选择的某种路由选择算法），或对外部隐藏其网络的内部组织面貌。在理想情况下，一个组织应当能够按自己的愿望运行和管理其网络，还要能将其网络与其他外部网络连接起来。

这两个问题都可以通过将路由器组织进**自治系统**（Autonomous System，AS）来解决，其中每个AS由一组通常处在相同管理控制下的路由器组成。通常在一个ISP中的路由器以及互联它们的链路构成一个AS。然而，某些ISP将它们的网络划分为多个AS。特别是，某些一级ISP在其整个网络中使用一个庞大的AS，而其他ISP则将它们的ISP拆分为数十个互联的AS。一个自治系统由其全局唯一的AS号（ASN）所标识［RFC 1930］。就像IP地址那样，AS号由ICANN区域注册机构所分配［ICANN 2016］。

在相同AS中的路由器都运行相同的路由选择算法并且有彼此的信息。在一个自治系统内运行的路由选择算法叫作**自治系统内部路由选择协议**（intra-autonomous system routing protocol）。

开放最短路优先（OSPF）

OSPF路由选择及其关系密切的协议IS-IS都被广泛用于因特网的AS内部路由选

择。OSPF 中的开放（open）一词是指路由选择协议规范是公众可用的（与之相反的是 Cisco 的 EIGRP 协议，该协议在最近才成为开放的 [Savage 2015]，它作为 Cisco 专用协议大约有 20 年时间）。OSPF 的最新版本是版本 2，由 [RFC 2328] 这个公用文档所定义。

OSPF 是一种链路状态协议，它使用洪泛链路状态信息和 Dijkstra 最低开销路径算法。使用 OSPF，一台路由器构建了一幅关于整个自治系统的完整拓扑图（即一幅图）。于是，每台路由器在本地运行 Dijkstra 的最短路径算法，以确定一个以自身为根节点到所有子网的最短路径树。各条链路开销是由网络管理员配置的（参见“实践原则：设置 OSPF 链路权值”）。管理员也许会选择将所有链路开销设为 1，因而实现了最少跳数路由选择，或者可能会选择将链路权值按与链路容量成反比来设置，从而不鼓励流量使用低带宽链路。OSPF 不强制使用设置链路权值的策略（那是网络管理员的任务），而是提供了一种机制（协议），为给定链路权值集合确定最低开销路径的路由选择。

使用 OSPF 时，路由器向自治系统内所有其他路由器广播路由选择信息，而不仅仅是向其相邻路由器广播。每当一条链路的状态发生变化时（如开销的变化或连接/中断状态的变化），路由器就会广播链路状态信息。即使链路状态未发生变化，它也要周期性地（至少每隔 30 min 一次）广播链路状态。RFC 2328 中有这样的说明：“链路状态通告的这种周期性更新增加了链路状态算法的健壮性。” OSPF 通告包含在 OSPF 报文中，该 OSPF 报文直接由 IP 承载，对 OSPF 其上层协议的值为 89。因此 OSPF 协议必须自己实现诸如可靠报文传输、链路状态广播等功能。OSPF 协议还要检查链路正在运行（通过向相连的邻居发送 HELLO 报文），并允许 OSPF 路由器获得相邻路由器的网络范围链路状态的数据库。

OSPF 的优点包括下列几方面：

- *安全*。能够鉴别 OSPF 路由器之间的交换（如链路状态更新）。使用鉴别，仅有受信任的路由器能参与一个 AS 内的 OSPF 协议，因此可防止恶意入侵者（或正在利用新学的知识到处试探的网络专业的学生）将不正确的信息注入路由器表内。在默认状态下，路由器间的 OSPF 报文是未被鉴别的并能被伪造。能够配置两类鉴别，即简单的和 MD5 的（参见第 8 章有关 MD5 和鉴别的一般性讨论）。使用简单的鉴别，每台路由器配置相同的口令。当一台路由器发送一个 OSPF 分组，它以明文方式包括了口令。显然，简单鉴别并不是非常安全。MD5 鉴别基于配置在所有路由器上的共享秘密密钥。对发送的每个 OSPF 分组，路由器对附加了秘密密钥的 OSPF 分组内容计算 MD5 散列值（参见第 8 章中报文鉴别码的讨论）。然后路由器将所得的散列值包括在该 OSPF 分组中。接收路由器使用预配置的秘密密钥计算出该分组的 MD5 散列值，并与该分组携带的散列值进行比较，从而验证了该分组的真实性。在 MD5 鉴别中也使用了序号对重放攻击进行保护。
- *多条相同开销的路径*。当到达某目的地的多条路径具有相同的开销时，OSPF 允许使用多条路径（这就是说，当存在多条相等开销的路径时，无须仅选择单一的路径来承载所有的流量）。
- *对单播与多播路由选择的综合支持*。多播 OSPF（MOSPF）[RFC 1584] 提供对 OSPF 的简单扩展，以便提供多播路由选择。MOSPF 使用现有的 OSPF 链路数据库，并为现有的 OSPF 链路状态广播机制增加了一种新型的链路状态通告。

- 支持在单个 AS 中的层次结构。一个 OSPF 自治系统能够层次化地配置多个区域。每个区域都运行自己的 OSPF 链路状态路由选择算法，区域内的每台路由器都向该区域内的所有其他路由器广播其链路状态。在每个区域内，一台或多台区域边界路由器负责为流向该区域以外的分组提供路由选择。最后，在 AS 中只有一个 OSPF 区域配置成主干区域。主干区域的主要作用是为该 AS 中其他区域之间的流量提供路由选择。该主干总是包含本 AS 中的所有区域边界路由器，并且可能还包含了一些非边界路由器。在 AS 中的区域间的路由选择要求分组先路由到一个区域边界路由器（区域内路由选择），然后通过主干路由到位于目的区域的区域边界路由器，进而再路由到最终目的地。

OSPF 是一个相当复杂的协议，而我们这里的讨论是十分简要的，［Huitema 1998；Moy 1998；RFC 2328］提供了更多的细节。

实践原则

设置 OSPF 链路权值

我们有关链路状态路由选择的讨论隐含地假设了下列事实：链路权重已经设置好了，运行诸如 OSPF 这样的路由选择算法，流量根据由 LS 算法计算所得的路由选择表流动。就因果而言，给定链路权重（即它们先发生），结果得到（经 Dijkstra 算法）最小化总体开销的路由选择路径。从这个角度看，链路权重反映了使用一条链路的开销（例如，如果链路权重与容量成反比，则使用高容量链路将具有较小的权重并因此从路由选择的角度更有吸引力），并且使用 Dijkstra 算法使得总开销为最小。

在实践中，链路权重和路由选择路径之间的因果关系也许是相反的，网络操作员配置链路权重，以获取某些流量工程目标的路由选择路径［Fortz 2000；Fortz 2002］。例如，假设某网络操作员具有在每个入口点进入和发向每个出口点的该网络的流量估计。该操作员接下来可能要设置特定入口到出口的流路由选择，以最小化经所有网络链路的最大利用率。但使用如 OSPF 这样的路由选择算法，操作员调节网络流的路由选择的主要手段就是链路权重。因此，为了取得最小化最大链路利用率的目标，操作员必须找出取得该目标的链路权重集合。这是一种相反的因果关系，即所希望的流路由选择已知，必须找到 OSPF 链路权重，使得该 OSPF 路由选择算法导致这种希望的流路由选择。

5.4　ISP 之间的路由选择：BGP

我们刚才学习了 OSPF 是一个 AS 内部路由选择协议。当在相同 AS 内的源和目的地之间进行分组选路时，分组遵循的路径完全由 AS 内路由选择协议所决定。然而，当分组跨越多个 AS 进行路由时，比如说从位于马里廷巴克图的智能手机到位于美国硅谷数据中心的一台服务器，我们需要一个**自治系统间路由选择协议**（inter-autonomous system routing protocol）。因为 AS 间路由选择协议涉及多个 AS 之间的协调，所以 AS 通信必须运行相同的 AS 间路由选择协议。在因特网中，所有的 AS 运行相同的 AS 间路由选择协议，称为**边界网关协议**（Broder Gateway Protocol，BGP）［RFC 4271；Stewart 1999］。

BGP 无疑是所有因特网协议中最为重要的（唯一竞争者可能是我们已经在 4.3 节中学

习的 IP 协议)，因为正是这个协议将因特网中数以千计的 ISP 黏合起来。如我们将看到的那样，BGP 是一种分布式和异步的协议，与 5.2.2 节中描述的距离向量路由选择协议一脉相承。尽管 BGP 是一种复杂和富有挑战性的协议，但为了深层次理解因特网，我们需要熟悉它的基础结构和操作。我们专注于学习 BGP 的时间将是物有所值的。

5.4.1 BGP 的作用

为了理解 BGP 的职责所在，考虑一个 AS 和在该 AS 中的任意一个路由器。前面讲过，每台路由器具有一张转发表，该转发表在将到达分组转发到出路由器链路的过程中起着主要作用。如我们已经学习过的那样，对于位于相同 AS 中的目的地而言，在路由器转发表中的表项由 AS 内部路由选择协议所决定。而对于位于该 AS 外部的目的地而言情况如何呢？这正是 BGP 用武之地。

在 BGP 中，分组并不是路由到一个特定的目的地址，相反是路由到 CIDR 化的前缀，其中每个前缀表示一个子网或一个子网的集合。在 BGP 的世界中，一个目的地可以采用 138.16.68/22 的形式，对于这个例子来说包括 1024 个 IP 地址。因此，一台路由器的转发表将具有形式为（x，I）的表项，其中 x 是一个前缀（例如 138.16.68/22），I 是该路由器的接口之一的接口号。

作为一种 AS 间的路由选择协议，BGP 为每台路由器提供了一种完成以下任务的手段：

1）*从邻居 AS 获得前缀的可达性信息。*特别是，BGP 允许每个子网向因特网的其余部分通告它的存在。一个子网高声宣布“我存在，我在这里”，而 BGP 确保在因特网中的所有 AS 知道该子网。如果没有 BGP 的话，每个子网将是隔离的孤岛，即它们孤独地存在，不为因特网其余部分所知和所达。

2）*确定到该前缀的“最好的”路由。*一台路由器可能知道两条或更多条到特定前缀的不同路由。为了确定最好的路由，该路由器将本地运行一个 BGP 路由选择过程（使用它经过相邻的路由器获得的前缀可达性信息)。该最好的路由将基于策略以及可达性信息来确定。

我们现在钻研 BGP 如何执行这两个任务。

5.4.2 通告 BGP 路由信息

考虑图 5-8 中显示的网络。如我们看到的那样，这个简单的网络具有 3 个自治系统：AS1、AS2 和 AS3。如显示的那样，AS3 包括一个具有前缀 x 的子网。对于每个 AS，每台路由器要么是一台**网关路由器**（gateway router），要么是一台**内部路由器**（internal router）。网关路由器是一台位于 AS 边缘的路由器，它直接连接到在其他 AS 中的一台或多台路由器。内部路由器仅连接在它自己 AS 中的主机和路由器。例如，在 AS1 中路由器 1c 是网关路由器；路由器 1a、1b 和 1d 是内部路由器。

我们考虑这样一个任务：向图 5-8 中显示的所有路由器通告对于前缀 x 的可达性信息。在高层次上，这是简明易懂的。首先，AS3 向 AS2 发送一个 BGP 报文，告知 x 存在并且位于 AS3 中；我们将该报文表示为“AS3 x”。然后 AS2 向 AS1 发送一个 BGP 报文，告知 x 存在并且能够先通过 AS2 然后进入 AS3 进而到达 x；我们将该报文表示为“AS2 AS3 x”。以这种方式，每个自治系统不仅知道 x 的存在，而且知道通向 x 的自治系统的路径。

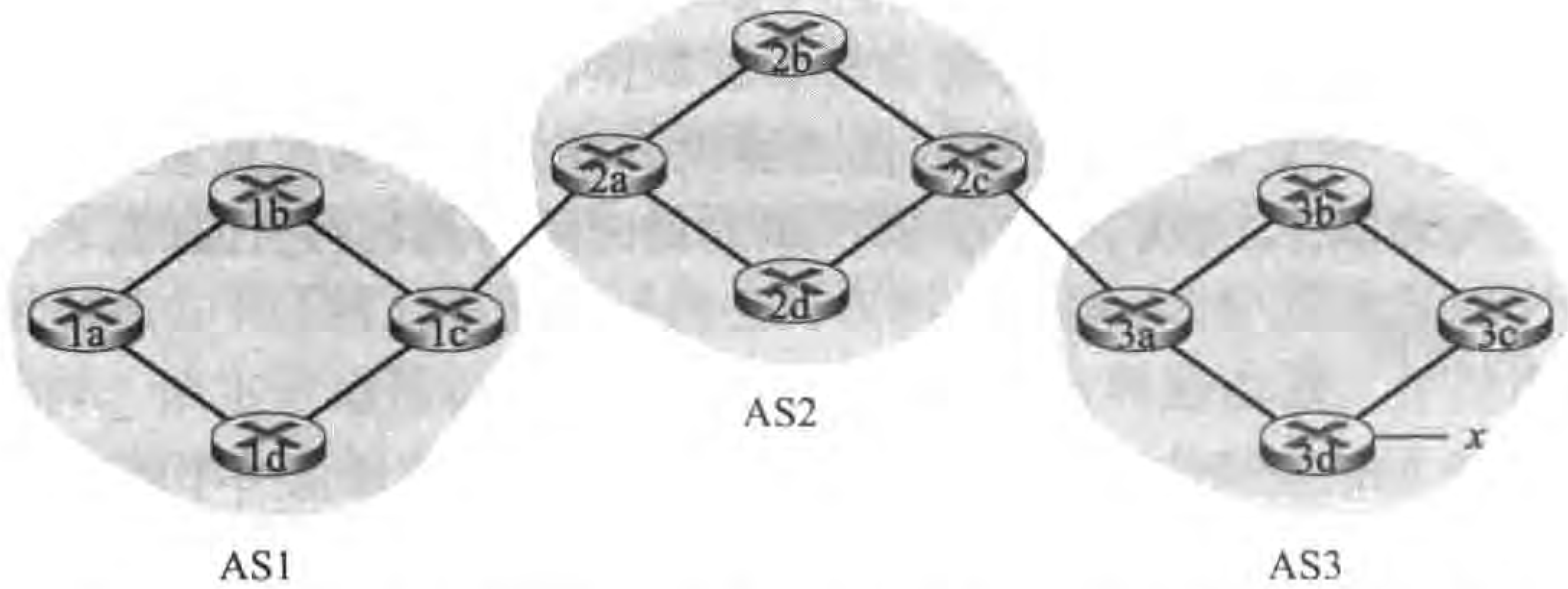


图 5-8 具有 3 个自治系统的网络。AS3 包括一个具有前缀 x 的子网

虽然在图 5-8 中有关通告 BGP 可达性信息的讨论能得到路径穿越的大意，但就自治系统彼此并未实际发送报文而言，它并不是准确的，相反是路由器在发送报文。为了理解这一点，我们现在重温图 5-8 中的例子。在 BGP 中，每对路由器通过使用 179 端口的半永久 TCP 连接交换路由选择信息。每条直接连接以及所有通过该连接发送的 BGP 报文，称为 **BGP 连接**（BGP connection）。此外，跨越两个 AS 的 BGP 连接称为**外部 BGP**（eBGP）连接，而在相同 AS 中的两台路由器之间的 BGP 会话称为**内部 BGP**（iBGP）连接。图 5-8 所示网络的 BGP 连接的例子显示在图 5-9 中。对于直接连接在不同 AS 中的网关路由器的每条链路而言，通常有一条 eBGP 连接；因此，在图 5-9 中，在网关路由器 1c 和 2a 之间有一条 eBGP 连接，而在网关路由器 2c 和 3a 之间也有一条 eBGP 连接。

在每个 AS 中的路由器之间还有多条 iBGP 连接。特别是，图 5-9 显示了一个 AS 内部的每对路由器之间的一条 BGP 连接的通常配置，在每个 AS 内部产生了网状的 TCP 连接。在图 5-9 中，eBGP 会话显示为长虚线，iBGP 显示为短虚线。注意到 iBGP 连接并不总是与物理链路对应。

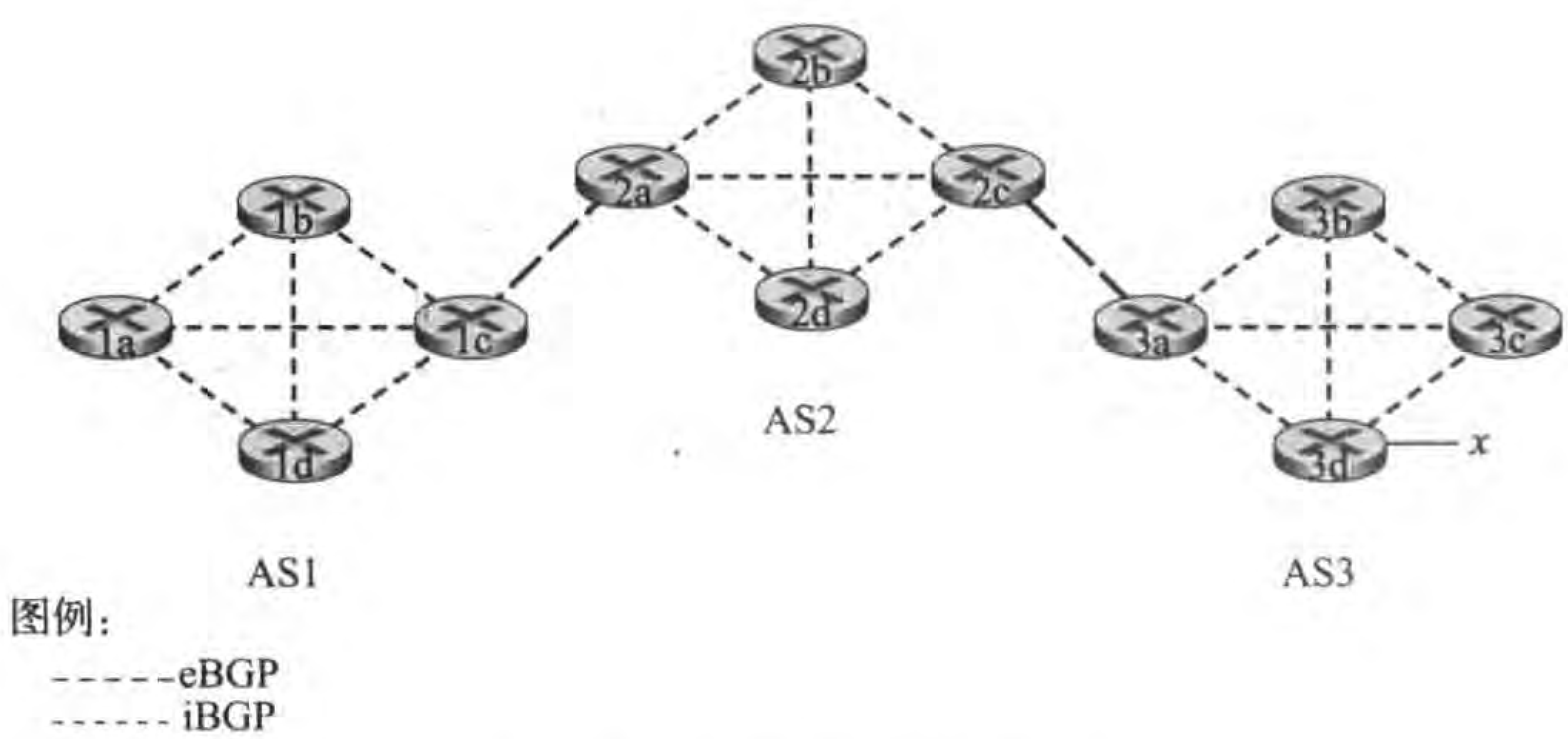


图 5-9 eBGP 和 iBGP 连接

为了传播可达性信息，使用了 iBGP 和 eBGP 会话。再次考虑向 AS1 和 AS2 中的所有路由器通告前缀 x 的可达性信息。在这个过程中，网关路由器 3a 先向网关路由器 2c 发送一个 eBGP 报文“AS3 x”。网关路由器 2c 然后向 AS2 中的所有其他路由器（包括网关路由器 2a）发送 iBGP 报文“AS3 x”。网关路由器 2a 接下来向网关路由器 1c 发送一个 eBGP 报文“AS2 AS3 x”。最后，网关路由器 1c 使用 iBGP 向 AS1 中的所有路由器发送报文“AS2 AS3 x”。在这个过程完成后，在 AS1 和 AS2 中的每个路由器都知道了 x 的存在并且也都知道了通往 x 的 AS 路径。

当然，在真实的网络中，从某个给定的路由器到某个给定的目的地可能有多条不同的路径，每条通过了不同的 AS 序列。例如，考虑图 5-10 所示的网络，它是在图 5-8 那个初始网络基础上，从路由器 1d 到路由器 3d 附加了一条物理链路。在这种情况下，从 AS1 到 x 有两条路径：经过路由器 1c 的路径 "AS2 AS3 x"；以及经过路由器 1d 的新路径 "AS3 x"。

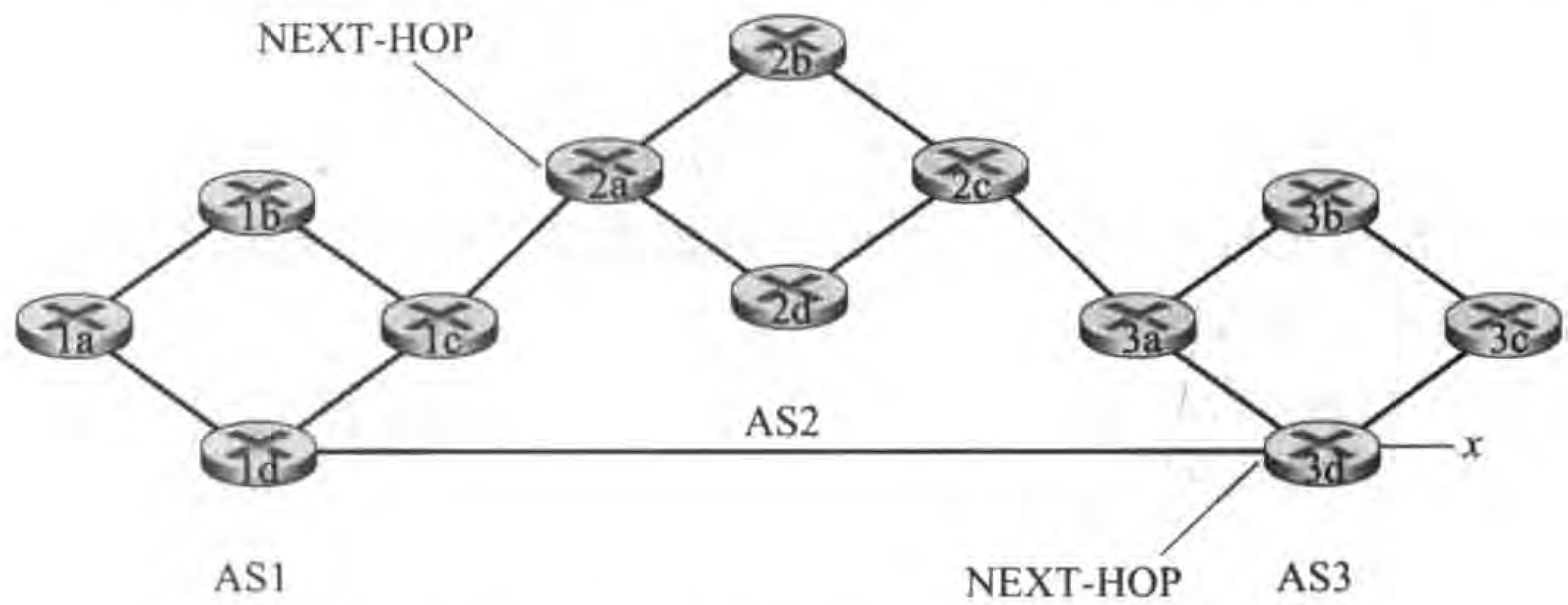


图 5-10　在 AS1 和 AS3 之间增加对等链路后的网络

5.4.3　确定最好的路由

如我们刚才学习到的那样，从一个给定的路由器到一个目的子网可能有多条路径。事实上，因特网中的路由器常常接收到很多不同的可能路径的可达性信息。一台路由器如何在这些路径之间进行选择（并且再相应地配置它的转发表）呢？

在处理这个关键性问题之前，我们需要引入几个 BGP 术语。当路由器通过 BGP 连接通告前缀时，它在前缀中包括一些 **BGP 属性**（BGP attribute）。用 BGP 术语来说，前缀及其属性称为**路由**（route）。两个较为重要的属性是 AS-PATH 和 NEXT-HOP。AS-PATH 属性包含了通告已经通过的 AS 的列表，如我们在前面的例子中所见。为了生成 AS-PATH 的值，当一个前缀通过某 AS 时，该 AS 将其 ASN 加入 AS-PATH 中的现有列表。例如，在图 5-10 中，从 AS1 到子网 x 有两条路：其中一条使用 AS-PATH "AS2 AS3"；而另一条使用 AS-PATH "AS3"。BGP 路由器还使用 AS-PATH 属性来检测和防止通告环路；特别是，如果一台路由器在路径列表中看到包含了它自己的 AS，它将拒绝该通告。

在 AS 间和 AS 内部路由选择协议之间提供关键链路方面，NEXT-PATH 属性具有敏感而重要的作用。NEXT-HOP 是 AS-PATH 起始的路由器接口的 IP 地址。为了深入理解该属性，我们再次参考图 5-10。如图 5-10 中所指示的那样，对于从 AS1 通过 AS2 到 x 的路由 "AS2 AS3 x"，其属性 NEXT-HOP 是路由器 2a 左边接口的 IP 地址。对于从 AS1 绕过 AS2 到 x 的路由 "AS3 x"，其 NEXT-HOP 属性是路由器 3d 最左边接口的 IP 地址。总的说来，在这个假想的例子中，AS1 中的每台路由器都知道了到前缀 x 的两台 BGP 路由：

　　路由器 2a 的最左侧接口的 IP 地址：AS2 AS3；x

　　路由器 3d 的最左侧接口的 IP 地址：AS3；x

这里，每条 BGP 路由包含 3 个组件：NEXT-HOP；ASPATH；目的前缀。在实践中，一条 BGP 路由还包括其他属性，眼下我们将暂且忽略它。注意到 NEXT-HOP 属性是不属于 AS1 的某路由器的 IP 地址；然而，包含该 IP 地址的子网直接连接到 AS1。

1. 热土豆路由选择

终于到了以精确的方式来讨论 BGP 路由选择算法的时刻了。我们将以一个最简单的路由选择算法开始，即**热土豆路由选择**（hot potato routing）。

考虑在图5-10网络中的路由器1b。如同刚才所述，这台路由器将学习到达前缀 x 的两条BGP路由。使用热土豆路由选择，（从所有可能的路由中）选择的路由到开始该路由的NEXT-HOP路由器具有最小开销。在这个例子中，路由器1b将查阅它的AS内部路由选择信息，以找到通往NEXT-HOP路由器2a的最低开销AS内部路径以及通往NEXT-HOP路由器3d的最低开销AS间路径，进而选择这些最低开销路径中具有最低开销的那条。例如，假设开销定义为穿越的链路数。则从路由器1b到路由器2a的最低开销是2，从路由器1b到路由器2d的最低开销是3，因此将选择路由器2a。路由器1b则将查阅它的转发表（由它的AS内部算法所配置），并且找到通往路由器2a的位于最低开销路径上的接口 I。1b则把（x，I）加到它的转发表中。

图5-11中总结了在一台路由器转发表中对于热土豆路由选择增加AS向外前缀的步骤。注意到下列问题是重要的：当在转发表中增加AS向外前缀时，AS间路由选择协议（BGP）和AS内部路由选择协议（如OSPF）都要用到。



图5-11 在路由器转发表中增加AS外部目的地的步骤

热土豆路由选择依据的思想是：对于路由器1b，尽可能快地将分组送出其AS（更明确地说，用可能的最低开销），而不担心其AS外部到目的地的余下部分的开销。就“热土豆路由选择”名称而言，分组被类比为烫手的热土豆。因为它烫手，你要尽可能快地将它传给另一个人（另一个AS）。热土豆路由选择因而是自私的算法，即它试图减小在它自己AS中的开销，而忽略在其AS之外的端到端开销的其他部分。注意到使用热土豆路由选择，对于在相同AS中的两台路由器，可能对相同的前缀选择两条不同的AS路径。例如，我们刚才看到路由器1b到达 x 将通过AS2发送分组。而路由器1d将绕过AS2并直接向AS3发送分组到达 x。

2. 路由器选择算法

在实践中，BGP使用了一种比热土豆路由选择更为复杂但却结合了其特点的算法。对于任何给定的目的地前缀，进入BGP的路由选择算法的输入是到某前缀的所有路由的集合，该前缀是已被路由器学习和接受的。如果仅有一条这样的路由，BGP则显然选择该路由。如果到相同的前缀有两条或多条路由，则顺序地调用下列消除规则直到余下一条路由：

1）路由被指派一个**本地偏好**（local preference）值作为其属性之一（除了AS-PATH和NEXT-HOP以外）。一条路由的本地偏好可能由该路由器设置或可能由在相同AS中的另一台路由器学习到。本地偏好属性的值是一种策略决定，它完全取决于该AS的网络管理员（我们随后将更为详细地讨论BGP策略问题）。具有最高本地偏好值的路由将被选择。

2）从余下的路由中（所有都具有相同的最高本地偏好值），将选择具有最短AS-PATH的路由。如果该规则是路由选择的唯一规则，则BGP将使用距离向量算法决定路径，其中距离测度使用AS跳的跳数而不是路由器跳的跳数。

3）从余下的路由中（所有都具有相同的最高本地偏好值和相同的AS-PATH长度），

使用热土豆路由选择，即选择具有最靠近 NEXT-HOP 路由器的路由。

4）如果仍留下多条路由，该路由器使用 BGP 标识符来选择路由，参见［Stewart 1999］。

举一个例子，我们再次考虑图 5-10 中的路由器 1b。前面讲过到前缀 x 确切地有两条 BGP 路由，一条通过 AS2 而另一条绕过 AS2。前面也讲过如果它使用自己的热土豆路由选择，则 BGP 将通过 AS2 向前缀 x 路由分组。但在上面的路由选择算法中，在规则 3 之前应用了规则 2，导致 BGP 选择绕过 AS2 的那条路由，因为该路由具有更短的 AS-PATH。因此我们看到使用上述路由选择算法，BGP 不再是一种自私的算法，即它先查找具有短 AS 路径的路由（因而很可能减小端到端时延）。

如上所述，BGP 是因特网 AS 间路由选择事实上的标准。要查看从第 1 层 ISP 中提取的各种 BGP 路由选择表（庞大！），可参见 http://www.routeviews.org。BGP 路由选择表通常包含超过 50 万条路由（即前缀和相应的属性）。BGP 路由选择表的规模和特征的统计可在［Potaroo 2016］中找到。

5.4.4 IP 任播

除了作为因特网的 AS 间路由选择协议外，BGP 还常被用于实现 IP 任播（anycast）服务［RFC 1546，RFC 7094］，该服务通常用于 DNS 中。为了说明 IP 任播的动机，考虑在许多应用中，我们对下列情况感兴趣：①在许多分散的不同地理位置，替换不同服务器上的相同内容；②让每个用户从最靠近的服务器访问内容。例如，一个 CDN 能够更换位于不同国家、不同服务器上的视频和其他对象。类似地，DNS 系统能够在遍及全世界的 DNS 服务器上复制 DNS 记录。当一个用户要访问该复制的内容，可以将用户指向具有该复制内容的“最近的”服务器。BGP 的路由选择算法为做这件事提供了一种最为容易和自然的机制。

为使我们的讨论具体，我们描述 CDN 可能使用 IP 任播的方式。如图 5-12 所示，在 IP 任播配置阶段，CDN 公司为它的多台服务器指派相同的 IP 地址，并且使用标准的 BGP 从这些服务器的每台来通告该 IP 地址。当某台 BGP 路由器收到对于该 IP 地址的多个路由通告，它将这些通告处理为对相同的物理位置提供不同的路径（事实上，这时这些通告对不同的物理位置是有不同路径的）。当配置其路由选择表时，每台路由器将本地化地使用 BGP 路由选择算法来挑选到该 IP 地址的“最好的”（例如，由 AS 跳计数确定的最近的）路由。例如，如果一个 BGP 路由（对应于一个位置）离该路由器仅一 AS 跳的距离，并且所有其他 BGP 路由（对应于其他位置）是两 AS 跳和更多 AS 跳，则该 BGP 路由器将选择把分组路由到一跳远的那个位置。在这个初始 BGP 地址通告阶段后，CDN 能够进行其分发内容的主要任务。当某客户请求视频时，CDN 向该客户返回由地理上分散的服务器所使用的共同 IP 地址，而无论该客户位于何处。当该客户想向那个 IP 地址发送一个请求时，因特网路由器则向那个“最近的”服务器转发该请求分组，最近的服务器是由 BGP 路由选择算法所定义的。

尽管上述 CDN 的例子很好地诠释了能够如何使用 IP 任播，但实践中 CDN 通常选择不使用 IP 任播，因为 BGP 路由选择变化能够导致相同的 TCP 连接的不同分组到达 Web 服务器的不同实例。但 IP 任播被 DNS 系统广泛用于将 DNS 请求指向最近的根 DNS 服务器。2.4 节讲过，当前根 DNS 服务器有 13 个 IP 地址。但对应于这些地址的每一个，有多个 DNS 根服务器，其中有些地址具有 100 多个 DNS 根服务器分散在世界的各个角落。当一个 DNS 请求向这 13 个 IP 地址发送时，使用 IP 任播将该请求路由到负责该地址的最近的那个 DNS 根服务器。

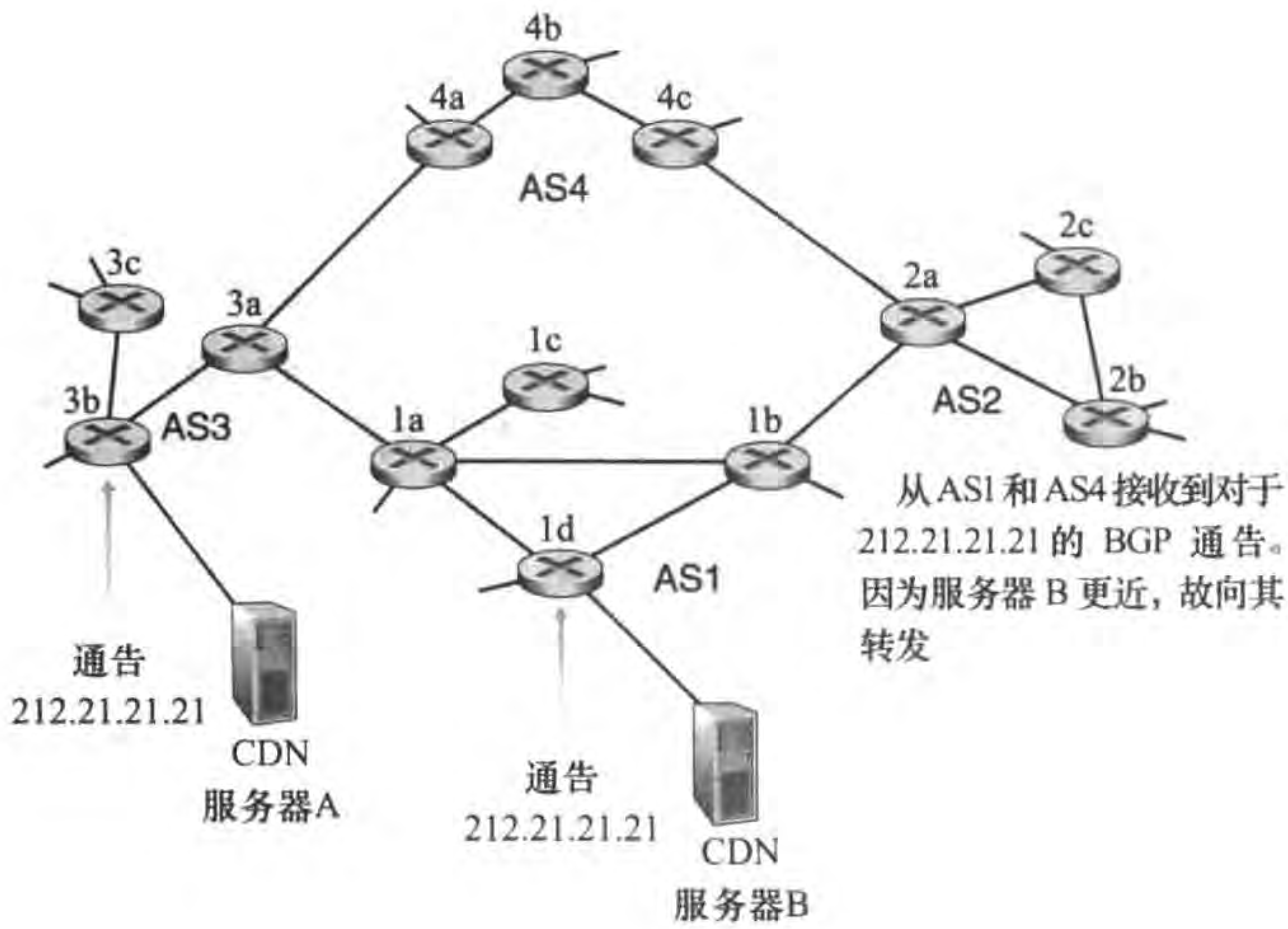


图 5-12 使用 IP 任播将用户引向最近的 CDN 服务器

5.4.5 路由选择策略

当某路由器选择到目的地的一条路由时，AS 路由选择策略能够胜过所有其他考虑，例如最短 AS 路径或热土豆路由选择。在路由选择算法中，实际上首先根据本地偏好属性选择路由，本地偏好值由本地 AS 的策略所确定。

我们用一个简单的例子说明 BGP 路由选择策略的某些基本概念。图 5-13 显示了 6 个互联的自治系统：A、B、C、W、X 和 Y。重要的是注意到 A、B、C、W、X 和 Y 是 AS，而不是路由器。假设自治系统 W、X 和 Y 是接入 ISP，而 A、B 和 C 是主干提供商网络。我们还要假设 A、B 和 C 直接向彼此发送流量，并向它们的客户网络提供全部的 BGP 信息。所有进入一个接入 ISP 网络的流量必定是以该网络为目的地，所有离开一个接入 ISP 网络的流量必定源于该网络。W 和 Y 显然是接入 ISP。X 是一个**多宿接入 ISP**（multi-homed stub network），因为它是经由两个不同的提供商连到网络的其余部分（这种方法在实践中变得越来越普遍）。然而，就像 W 和 Y 一样，X 自身必定是进入/离开 X 的所有流量的源/目的地。但这种桩网络的行为是如何实现和强制实现的呢？X 如何防止转发 B 与 C 之间的流量呢？这能够通过控制 BGP 路由的通告方式容易地实现。特别是，X 如果（向其邻居 B 和 C）通告它没有通向（除自身以外）任何其他目的地的路径，那么它将起到一个接入 ISP 的作用。这就是说，即使 X 可能知道一条路径（比如说 XCY）能到达网络 Y，它也将不把该条路径通告给 B。由于 B 不知道 X 有一条路径到 Y，B 绝不会经由 X 转发目的为 Y（或 C）的流量。这个简单的例子说明了如何使用一条选择的路由通告策略来实现客户/提供商路由选择关系。

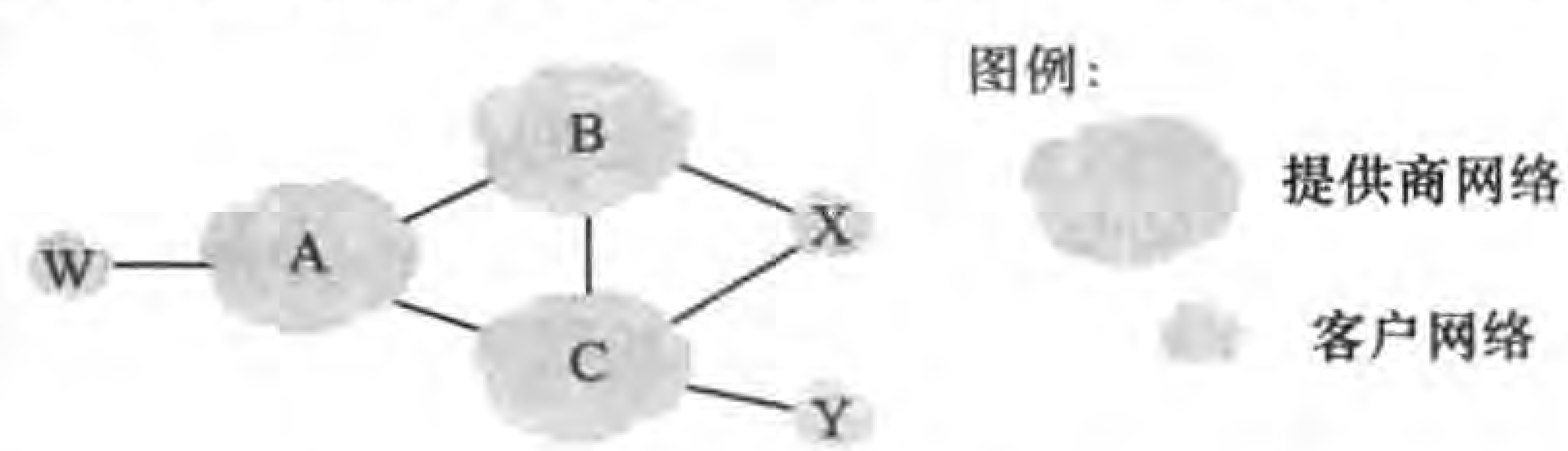


图 5-13 一个简单的 BGP 策略场景

我们接下来关注一个提供商网络，比如自治系统 B。假定 B 已经（从 A 处）知道了 A 有一条到 W 的路径 AW。B 因此能将路由 AW 安装到其路由信息库中。显然，B 也想向它的客户 X 通告路径 BAW，这样 X 知道它能够通过 B 路由到 W。但是，B 应该将路径 BAW

通告给 C 吗？如果它这样做，则 C 可以经由 BAW 将流量引导到 W。如果 A、B 和 C 都是主干提供商，而 B 也许正好觉得它不应该承担在 A 与 C 之间传送流量的负担（和开销）。B 可能有理由认为，确保 C 能经过 A 和 C 之间的直接连接引导 A 客户的来去流量是 A 和 C 的工作（和开销）。目前还没有强制主干 ISP 之间如何路由选择的官方标准。然而，商业运行的 ISP 们都遵从的一个经验法则是：任何穿越某 ISP 主干网的流量必须是其源或目的（或两者）位于该 ISP 的某个客户网络中；不然的话这些流量将会免费搭车通过该 ISP 的网络。各个对等协定（用于解决前面提到的问题）通常都是 ISP 双方进行协商，而且经常是对外保密的；[Huston 1999a] 提供了关于对等协定的有趣讨论。路由选择策略如何反映 ISP 之间的商业关系的详细描述参见 [Gao 2001；Dmitiropoulos 2007]。从 ISP 的立场出发，有关 BGP 路由选择策略的讨论参见 [Caesar 2005b]。

我们完成了对 BGP 的简要介绍。理解 BGP 是重要的，因为它在因特网中起着重要作用。我们鼓励你阅读参考文献 [Griffin 2012；Stewart 1999；Labovitz 1997；Halabi 2000；Huitema 1998；Gao 2001；Feamster 2004；Caesar 2005b；Li 2007]，以学习更多的 BGP 知识。

实践原则

为什么会有不同的 AS 间和 AS 内部路由选择协议？

学习了目前部署在因特网中的特定的 AS 间和 AS 内部路由选择协议的细节后，我们可通过思考对这些协议首先会问的也许最为根本性的问题来得到结论（希望你已经在思考该问题，并且不致因技术细节而不能把握全局）：为什么所使用的 AS 间和 AS 内部路由选择协议是不同的？

对该问题的答案触及了 AS 内与 AS 间的路由选择目标之间差别的本质：

- 策略。在 AS 之间，策略问题起主导作用。一个给定 AS 产生的流量不能穿过另一个特定的 AS，这可能非常重要。类似地，一个给定 AS 也许想很好地控制它承载的其他 AS 之间穿越的流量。我们已看到，BGP 承载了路径属性，并提供路由选择信息的受控分布，以便能做出这种基于策略的路由选择决策。在一个 AS 内部，一切都是在相同的管理控制名义下进行的，因此策略问题在 AS 内部选择路由中起着微不足道的作用。
- 规模。扩展一个路由选择算法及其数据结构以处理到大量网络或大量网络之间的路由选择的这种能力，是 AS 间路由选择的一个关键问题。在一个 AS 内，可扩展性不是关注的焦点。首先，如果单个 ISP 变得太大时，总是能将其分成两个 AS，并在这两个新的 AS 之间执行 AS 间路由选择。（前面讲过，OSPF 通过将一个 AS 分成区域而建立这样的层次结构。）
- 性能。由于 AS 间路由选择是面向策略的，因此所用路由的质量（如性能）通常是次要关心的问题（即一条更长或开销更高但能满足某些策略条件的路由也许被采用了，而更短但不满足那些条件的路由却不会被采用）。我们的确看到了在 AS 之间，甚至没有与路由相关的开销（除了 AS 跳计数外）概念。然而在一个 AS 内部，这种对策略的关心就不重要了，可以使路由选择更多地关注一条路由实现的性能级别。

5.4.6 拼装在一起：在因特网中呈现

尽管本小节不是有关BGP本身的，但它将我们到此为止看到的许多协议和概念结合到一起，包括IP地址、DNS和BGP。

假定你只是创建了一个具有若干服务器的小型公司网络，包括一台描述公司产品和服务的公共Web服务器，一台从你的雇员获得他们的电子邮件报文的电子邮件服务器和一台DNS服务器。你当然乐意整个世界能够访问你的Web站点，以得知你的现有产品和服务。此外，你将乐意你的雇员能够向遍及世界的潜在客户发送和接收电子邮件。

为了满足这些目标，你首先需要获得因特网连接，要做到这一点，需要与本地ISP签订合同并进行连接。你的公司将有一台网关路由器，该路由器将与本地ISP的一台路由器相连。该连接可以是一条通过现有电话基础设施的DSL连接、一条到ISP路由器的租用线，或者是第1章描述的许多其他接入解决方案之一。你的本地ISP也将为你提供一个IP地址范围，例如由256个地址组成的一个/24地址范围。一旦你有了自己的物理连接和IP地址范围，你将在该地址范围内分配IP地址：一个给你的Web服务器，一个给你的电子邮件服务器，一个给你的DNS服务器，一个给你的网关路由器，并将其他IP地址分配给公司网络中的其他服务器和联网设备。

除了与一个ISP签订合同外，你还需要与一个因特网注册机构签订合同，以便为你的公司获得一个域名，如在第2章中所描述的那样。例如，如果你的公司名称比如说是Xanadu Inc.，你自然希望获得域名xanadu.com。你的公司还必须呈现在DNS系统中。具体而言，因为外部世界将要联系你的DNS服务器以获得该服务器的IP地址，所以你还需要为注册机构提供你的DNS服务器的IP地址。该注册机构则在.com顶级域名服务器中为你的DNS服务器设置一个表项（域名和对应的IP地址），如第2章所述。在这个步骤完成后，任何知道你的域名（例如xanadu.com）的用户将能够经过DNS系统获得你DNS服务器的IP地址。

为了使人们能够发现你的Web服务器的IP地址，你需要在你的DNS服务器中包括一个将你的Web服务器的主机名（例如www.xanadu.com）映射到它的IP地址的表项。你还要为公司中其他公共可用的服务器设置类似的表项，包括你的电子邮件服务器。如此一来，如果Alice要浏览你的Web服务器，DNS系统将联系你的DNS服务器，找到你的Web服务器的IP地址，并将其给Alice。Alice则能与你的Web服务器创建一个直接的TCP连接。

然而，允许来自世界各地的外部人员访问你的Web服务器，仍然还有一个必要的、决定性的步骤。考虑当Alice做下列事情发生的状况：Alice知道你的Web服务器的IP地址，她向该IP地址发送一个IP数据报（例如一个TCP SYN报文段）。该数据报将通过因特网进行路由，经历了在许多不同的自治系统中的一系列路由器，最终到达你的Web服务器。当任何一个路由器收到该数据报时，将去它的转发表中寻找一个表项来确定转发该数据报的外出端口。因此，每台路由器需要知道你公司的/24前缀（或者某些聚合项）。一台路由器如何知道你公司的前缀呢？如我们刚才看到的那样，它从BGP知道了该前缀。具体而言，当你的公司与本地ISP签订合同并且获得了分配的前缀（即一个地址范围），你的本地ISP将使用BGP向与之连接的ISP通告你的前缀。这些ISP将依次使用BGP来传播该通告。最终，所有的因特网路由器将得知了你的前缀（或者包括你的前缀的某个聚合项），因而能够将数据报适当地转发到适当的Web和电子邮件服务器。

5.5 SDN 控制平面

在本节中，我们将深入 SDN 控制平面，即控制分组在网络的 SDN 使能设备中转发的网络范围逻辑，以及这些设备和它们的服务的配置与管理。这里的学习建立在前面 4.4 节中一般化 SDN 转发讨论的基础上，因此你在继续学习前需要先回顾一下那一节，以及本章的 5.1 节。如同 4.4 节中一样，我们将再次采用在 SDN 文献中所使用的术语，将网络的转发设备称之为“分组交换机”（或直接称为交换机，理解时带上“分组”二字），因为能够根据网络层源/目的地址、链路层源/目的地址以及运输层、网络层和链路层中分组首部字段做出转发决定。

SDN 体系结构具有 4 个关键特征 [Kreutz 2015]：

- *基于流的转发*。SDN 控制的交换机的分组转发工作，能够基于运输层、网络层或链路层首部中任意数量的首部字段值进行。在 4.4 节中，我们看到了 OpenFlow 1.0 抽象允许基于 11 个不同的首部字段值进行转发。这与我们 5.2 ~ 5.4 节中学习的基于路由器转发的传统方法形成了鲜明的对照，传统方法中 IP 数据报的转发仅依据数据报的目的 IP 地址进行。回顾图 5-2，分组转发规则被精确规定在交换机的流表中；SDN 控制平面的工作是计算、管理和安装所有网络交换机中的流表项。
- *数据平面与控制平面分离*。这种分离明显地显示在图 5-2 和图 5-14 中。数据平面由网络交换机组成，交换机是相对简单（但快速）的设备，该设备在它们的流表中执行“匹配加动作”的规则。控制平面由服务器以及决定和管理交换机流表的软件组成。
- *网络控制功能：位于数据平面交换机外部*。考虑到 SDN 中的“S”表示“软件”，也许 SDN 控制平面由软件实现并不令人惊讶。然而，与传统的路由器不同，这个软件在服务器上执行，该服务器与网络交换机截然分开且与之远离。如在图 5-14 中所示，控制平面自身由两个组件组成：一个 SDN 控制器（或网络操作系统 [Gude 2008]），以及若干网络控制应用程序。控制器维护准确的网络状态信息（例如，远程链路、交换机和主机的状态）；为运行在控制平面中的网络控制应用程序提供这些信息；提供方法，这些应用程序通过这些方法能够监视、编程和控制下面的网络设备。尽管在图 5-14 中的控制器显示为一台单一的服务器，但实践中控制器仅是逻辑上集中的，通常在几台服务器上实现，这些服务器提供协调的、可扩展的性能和高可用性。
- *可编程的网络*。通过运行在控制平面中的网络控制应用程序，该网络是可编程的。这些应用程序代表了

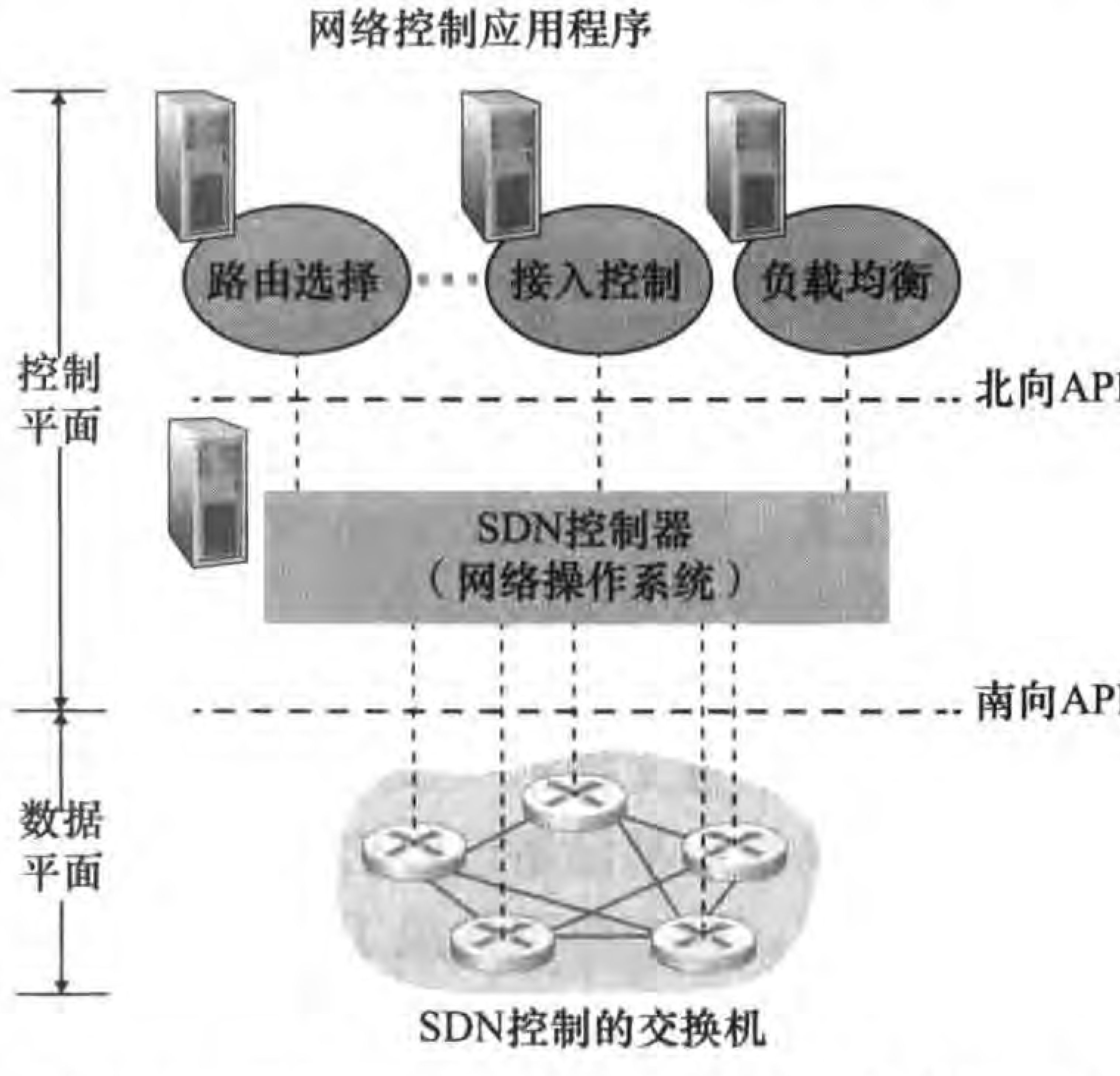


图 5-14 SDN 体系结构的组件：SDN 控制的交换机、SDN 控制器和网络控制应用程序

SDN 控制平面的“智力”，使用了由 SDN 控制器提供的 API 来定义和控制网络设备中的数据平面。例如，一个路由选择网络控制应用程序可以决定源和目的地之间的端到端路径（例如，通过使用由 SDN 控制器维护的节点状态和链路状态信息，执行 Dijkstra 算法）。另一个网络应用程序可以执行访问控制，即决定交换机阻挡哪个分组，如4.4.3 节中的第三个例子那样。还有一个应用程序可以用执行服务器负载均衡的方式转发分组（4.4.3 节中我们考虑的第二个例子）。

从讨论中我们可见，SDN 表示了一种意义重大的网络功能的“分类”，即数据平面交换机、SDN 控制器和网络控制应用程序是分离的实体，该实体可以由不同的厂商和组织机构所提供。这与 SDN 之前模式形成了鲜明对照，在 SDN 之前模式中，交换机/路由器（连同其嵌入的控制平面软件和协议实现）是一个整体，它是垂直、综合的，并且由单一的厂商所销售。在 SDN 中的这种网络功能分类，可以与大型计算机到个人计算机的早期演化相比拟，前者的硬件、系统软件和应用程序是由单一厂商所提供的，而后者具有各自的硬件、操作系统和应用程序。计算硬件、系统软件和应用程序的分类，无疑已经在所有这三个领域的创新驱动下导致了丰富、开放的生态系统。对 SDN 的希望是，它也将导致如此丰富的创新。

给出了我们对图 5-14 的 SDN 体系结构的理解，自然会产生许多问题。流表是如何实际计算的，以及在哪里进行的？响应 SDN 控制的设备产生的事件时（例如，一条附属链路的激活/关闭），这些流表是如何更新的呢？在协作的多台交换机中，流表项是如何以一种导致和谐、一致的网络范围功能的方式进行协作的呢（例如，用于转发分组的从源到目的地的端到端路径，或与分布式防火墙协作）？提供这些以及许多其他能力是 SDN 控制平面的作用。

5.5.1　SDN 控制平面：SDN 控制器和 SDN 网络控制应用程序

我们通过考虑控制平面必须提供的一般能力开始抽象地讨论 SDN 控制平面。如我们所见，这种抽象即“基本原理”方法将我们引向一个总体的体系结构，该体系结构反映了 SDN 控制平面如何在实际中实现。

如上所述，SDN 控制平面大体划分为两个部分，即 SDN 控制器和 SDN 网络控制应用程序。我们先来仔细考察控制器。自从最早的 SDN 控制器［Gude 2008］开发以来，已经研制了多种 SDN 控制器，文献［Kreutz 2015］极其全面地综述了最新进展。图 5-15 提供了一个通用SDN 控制器的更为详尽的视图。控制器的功能可大体组织为3 个层次。我们以一种非典型的自底向上方式考虑这些层次：

- *通信层：SDN 控制器和受控网络设备之间的通信。*显然，如果 SDN 控制器要控制远程 SDN 使能的交换机、主机或其他设备的运行，需要一个协议来传送控制器与这些设备之间的信息。此外，设备必须能够向控制器传递本地观察到的事件（例如，一个报文指示一条附属链路已经激活或停止，一个设备刚刚加入了网络，或一个心跳指示某设备已经启动和运行）。这些事件向 SDN 控制器提供该网络状态的最新视图。这个协议构成了控制器体系结构的最底层，如图 5-15 中所示。控制器和受控设备之间的通信跨越了一个接口，它现在被称为控制器的“南向”接口。在5.5.2 节中，我们将学习 OpenFlow，它是一种提供这种通信功能的特定协议。OpenFlow 在大多数 SDN 控制器中得到了实现（即使不是全部）。
- *网络范围状态管理层。*由 SDN 控制平面所做出的最终控制决定（例如配置所有交

换机的流表以取得所希望的端到端转发，实现负载均衡，或实现一种特定的防火墙能力)，将要求控制器具有有关网络的主机、链路、交换机和其他 SDN 控制设备的最新状态信息。交换机的流表包含计数器，其值也可以由网络控制应用程序很好地使用；因此这些值应当为应用程序所用。既然控制平面的终极目标是决定用于各种受控设备的流表，控制器也就可以维护这些表的拷贝。这些信息都构成了由 SDN 控制器维护的网络范围“状态”的例子。

- *对于网络控制应用程序层的接口*。控制器通过它的“北向”接口与网络控制应用程序交互。该 API 允许网络控制应用程序在状态管理层之间读/写网络状态和流表。当状态改变事件出现时，应用程序能够注册进行通告。可以提供不同类型的 API，我们将看到两种流行 SDN 控制器使用 REST [Fielding 2000] 请求响应接口与它们的应用程序进行通信。

我们已经提到过几次，SDN 控制器被认为是“逻辑上集中”的，即该控制器可以被外部视为一个单一、整体的服务（例如，从 SDN 控制设备和外部的网络控制应用程序的角度看)。然而，出于故障容忍、高可用性或性能等方面的考虑，在实践中这些服务和用于保持状态信息的数据库一般通过分布式服务器集合实现。在服务器集合实现控制器功能时，必须考虑控制器的内部操作（例如维护事件的逻辑时间顺序、一致性、意见一致等）的语义 [Panda 2013]。这些关注点在许多不同的分布式系统中都是共同的，这些挑战的简洁解决方案可参见 [Lamport 1989；Lampson 1996]。诸如 OpenDaylight 和 ONOS 这样的现代控制器已经将很大的注意力放在构建一种逻辑上集中但物理上分布的控制器平台上，该平台提供可扩展的服务和对受控设备以及网络控制应用程序的高可用性。

在图 5-15 中描述的体系结构与 2008 年最初提出的 NOX 控制器 [Gude 2008] 以及今天的 OpenDaylight [OpenDaylight Lithium 2016] 和 ONOS [ONOS 2016] SDN 控制器（参见插入材料）的体系结构极为相似。我们将在 5.5.3 节介绍一个控制器操作的例子。然而，我们先来仔细审视 OpenFlow 协议，该协议位于控制器的通信层中。

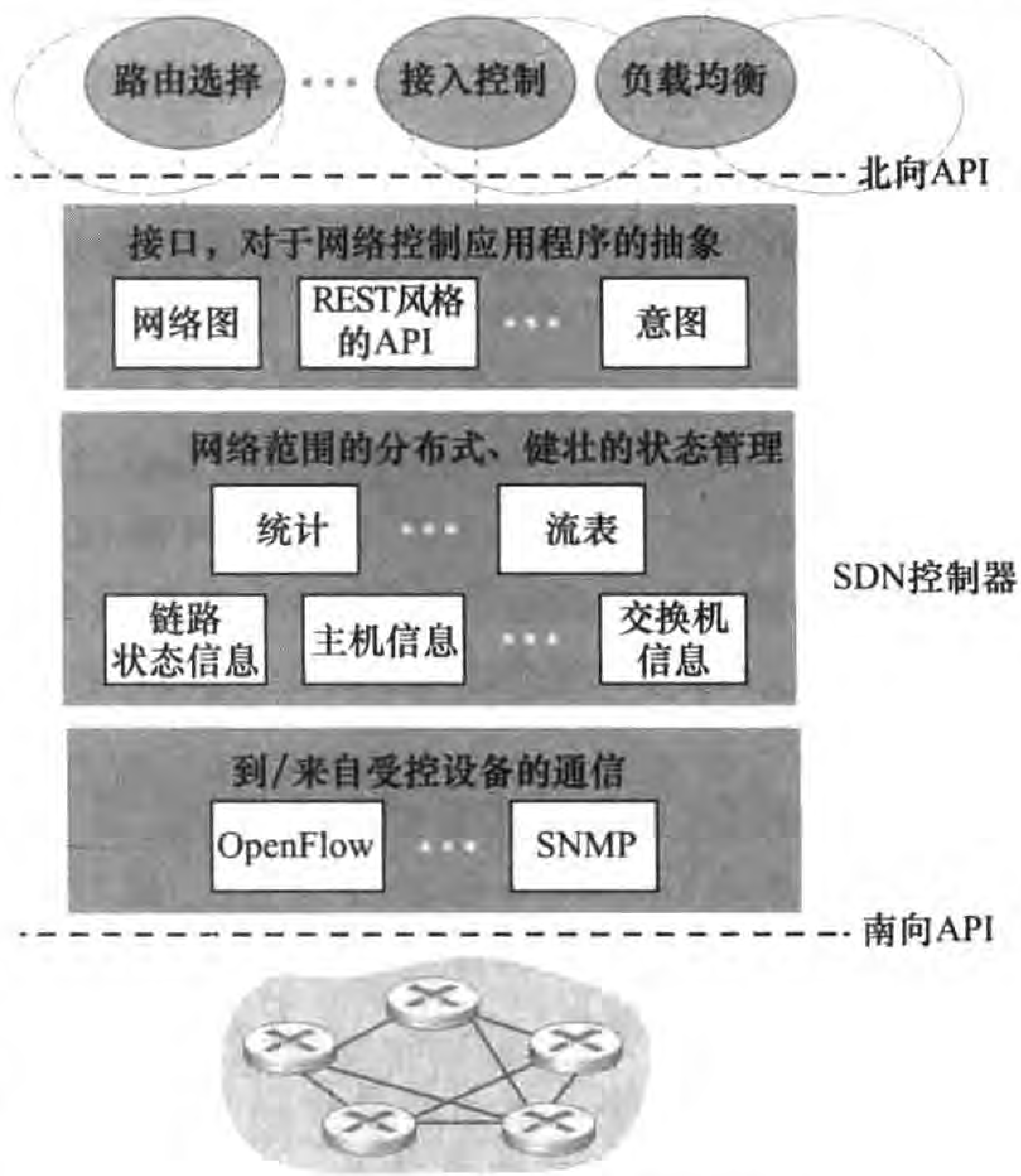


图 5-15　SDN 控制器的组件

5.5.2 OpenFlow 协议

OpenFlow 协议 [OpenFlow 2009；ONF 2016] 运行在 SDN 控制器和 SDN 控制的交换机或其他实现 OpenFlow API 的设备之间（OpenFlow API 我们在 4.4 节学习过）。OpenFlow 协议运行在 TCP 之上，使用 6653 的默认端口号。从控制器到受控交换机流动的重要报文有下列这些：

- 配置。该报文允许控制器查询并设置交换机的配置参数。
- 修改状态。该报文由控制器所使用，以增加/删除或修改交换机流表中的表项，并且设置交换机端口特性。
- 读状态。该报文被控制器用于从交换机的流表和端口收集统计数据和计数器值。
- 发送分组。该报文被控制器用于在受控交换机从特定的端口发送出一个特定的报文。

从受控交换机到控制器流动的重要报文有下列这些：

- 流删除。该报文通知控制器已删除一个流表项，例如由于超时，或作为收到“修改状态”报文的结果。
- 端口状态。交换机用该报文向控制器通知端口状态的变化。
- 分组入。4.4 节讲过，一个分组到达交换机端口，并且不能与任何流表项匹配，那么这个分组将被发送给控制器进行额外处理。匹配的分组也被发送给控制器，作为匹配时所采取的一个动作。“分组入”报文被用于将分组发送给控制器。另外的 OpenFlow 报文定义在［OpenFlow 2009；ONF 2016］中。

实践原则

谷歌的软件定义全球网络

2.6 节中的学习案例中讲过，谷歌部署了专用的广域网（WAN），该网互联了它（在 IXP 和 ISP 中）的数据中心和服务器集群。这个称为 B4 的网络有一个谷歌基于 OpenFlow 设计的 SDN 控制平面。谷歌网络能够在长途线路上以接近 70% 的利用率运行 WAN 链路（超过典型的链路利用率的 2～3 倍），并且基于应用优先权和现有的流需求在多条路径之间分割应用流［Jain 2013］。特别是，谷歌 B4 网络适合于 SDN：①从IXP 和 ISP 中的边缘服务器到网络核心中的路由器，谷歌控制了所有的设备；②带宽最密集的应用是场点之间的大规模数据拷贝，这种数据拷贝在资源拥塞期间能够“服从”较高优先权的交互应用；③由于仅连接了几十个数据中心，集中式控制是可行的。

谷歌的 B4 网络使用定制的交换机，每台交换机实现了 OpenFlow 稍加扩充的版本，具有一个本地 OpenFlow 代理（OFA），该 OFA 方法上类似于我们在图 5-2 中遇到的控制代理。每个 OFA 又与网络控制服务器（NCS）中的 OpenFlow 控制器（OFC）相连，使用一个分离的“带外”网络，与在数据中心之间承载数据中心流量的网络截然不同。该 OFC 因此提供由 NCS 使用的服务以与它的受控交换机通信，方法上类似于显示在图 5-15 中的 SDN 体系结构的最低层。在 B4 中，OFC 也执行状态管理功能，在网络信息库（NIB）中保持节点和链路状态。OFC 的谷歌实现基于 ONIX SDN 控制器［Koponen 2010］。实现了两种路由选择协议：用于数据中心之间路由选择的 BGP 和用于数据中心内部路由选择的 IS-IS（非常接近 OSPF）。Paxos［Chandra 2007］用于执行 NCS 组件的热复制，以防止故障。

一种逻辑上置于网络控制服务器集合之上的流量工程网络控制应用，与这些服务器交互，以为一组应用流提供全局、网络范围的带宽。借助于 B4，SDN 一举跨入全球网络提供商的运行网络的行列。B4 的详细描述请参见［Jain 2013］。

5.5.3 数据平面和控制平面交互的例子

为了具体地理解 SDN 控制的交换机与 SDN 控制器之间的交互，我们考虑图 5-16 中所示的例子，其中使用了 Dijkstra 算法（该算法我们已在 5.2 节中学习过）来决定最短路径路由。图 5-16 中的 SDN 场景与前面 5.2.1 节和 5.3 节中描述的每路由器控制场景有两个重要差异，Dijkstra 算法是实现在每台路由器中并且在所有网络路由器中洪泛链路状态更新：

- Dijkstra 算法作为一个单独的程序来执行，位于分组交换机的外部。
- 分组交换机向 SDN 控制器发送链路更新并且不互相发送。

在这个例子中，我们假设交换机 s1 和 s2 之间的链路断开；实现了最短路径路由选择，因此，除了 s2 操作未改变外，s1、s3 和 s4 的入和出流转发规则都受到影响。我们也假定 OpenFlow 被用作通信层协议，控制平面只执行链路状态路由选择而不执行其他功能。

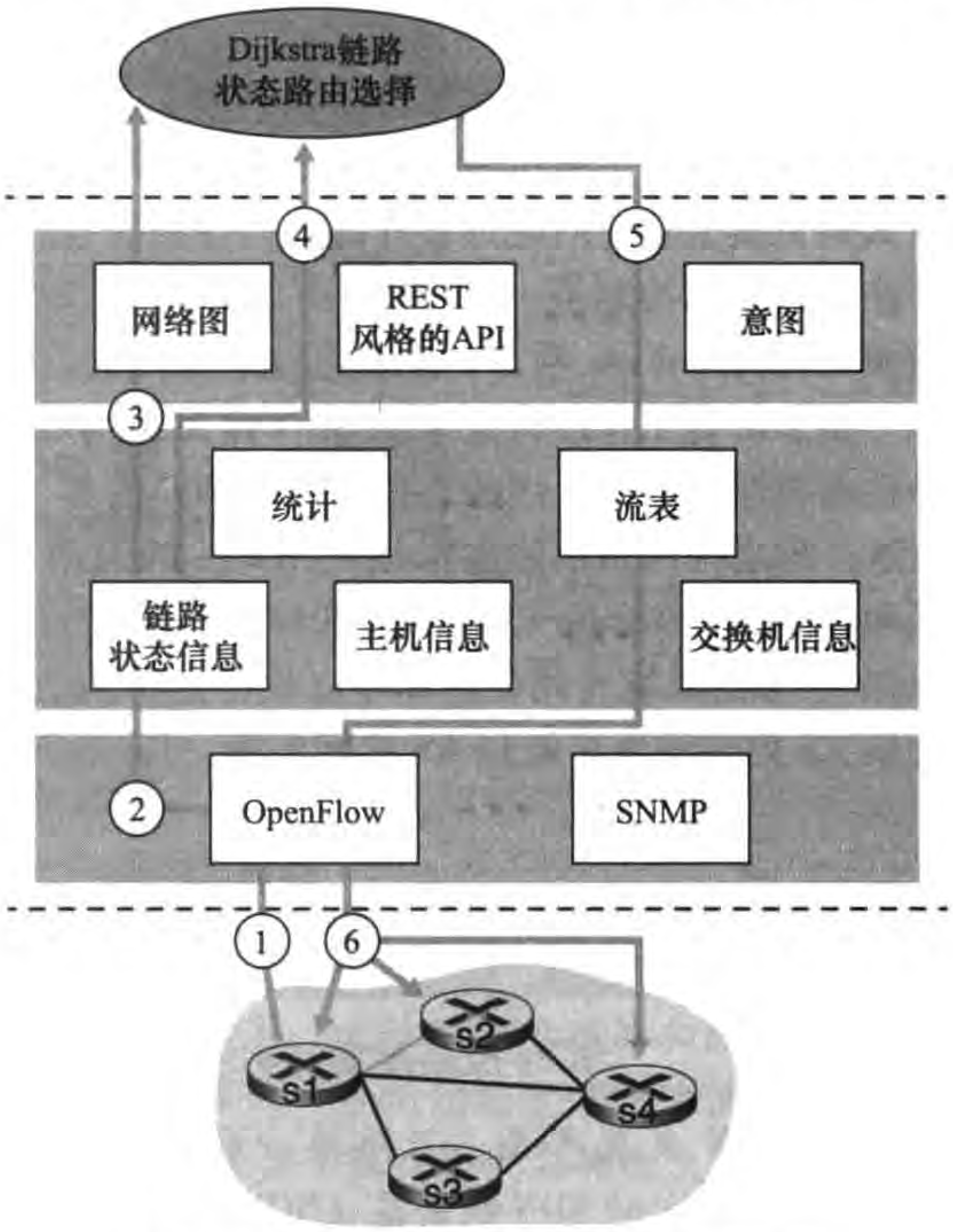


图 5-16 SDN 控制器场景：链路状态更新

1）交换机 s1 经历了自己与 s2 之间的链路故障，使用 OpenFlow “端口状态” 报文向 SDN 控制器通报该链路状态的更新。

2）SDN 控制器接收指示链路状态更新的 OpenFlow 报文，并且通告链路状态管理器，由管理器更新链路状态库。

3）实现 Dijkstra 链路状态路由选择的网络控制应用程序先前进行了注册，当链路状态更新时将得到通告。应用程序接收该链路状态更新的通告。

4）链路状态路由选择应用程序与链路状态管理器相互作用，以得到更新的链路状态；它也会参考状态管理层中的其他组件。然后它计算新的最低开销路径。

5）链路状态路由选择应用则与流表管理器交互，流表管理器决定更新的流表。

6）流表管理器则使用 OpenFlow 协议更新位于受影响的交换机 s1、s2 和 s4 的流表项，其中 s1 此时将经 s4 将分组的目的地指向 s2，s2 此时将经中间交换机 s4 开始接收来自 s1 的分组，s4 此时必须转发来自 s1 且目的地为 s2 的分组。

这个例子虽简单，但图示了 SDN 控制平面如何提供控制平面服务（此时为网络层路由选择），而该服务以前是以每路由器控制在每台路由器中实现的。我们现在能够容易地体会到，SDN 使能的 ISP 能够容易地将最低开销路径的路由选择转变为更加定制的路由选择方法。因为控制器的确能够随心所欲地定制流表，因此能够实现它喜欢的任何形式的转发，即只是通过改变它的应用控制软件。这种改变的便利性与传统的每路由器控制平面的

情况形成对照，传统的情况必须要改变所有路由器中的软件，而这些路由器可能是由多个不同厂商提供给 ISP 的。

5.5.4 SDN 的过去与未来

尽管对 SDN 的强烈兴趣是相对近期的现象，但 SDN 的技术根源，特别是数据平面和控制平面的分离，可追溯到相当久远。在 2004 年，文献［Feamster 2004；Lakshman 2004；RFC 3746］中都赞成网络数据平面与控制平面分离。［van der Merwe 1998］描述了用于具有多个控制器的 ATM 网络［Black 1995］的控制框架，每台控制器控制若干 ATM 交换机。Ethane 项目开拓了简单基于流的多台以太网交换机和一台集中式控制器的网络概念，其中以太网交换机具有匹配加动作流表，控制器管理流准入和路由选择，而未匹配的分组将从交换机转发到控制器。在 2007 年，超过 300 台 Ethane 交换机的网络投入运行。Ethane 迅速演化为 OpenFlow 项目，而其他的成为历史！

很多研究工作以研发未来 SDN 体系结构和能力为目标。如我们所见，SDN 革命正在导致颠覆性地替代专用的整体交换机和路由器（它们同时具有数据平面和控制平面）。类似地，称之为网络功能虚拟化（NFV）的通用 SDN 的目标是用简单的商用服务器、交换机和存储器［Gember-Jacobson 2014］来颠覆性地替代复杂的中间盒（例如用于媒体高速缓存/服务的具有专用硬件和专有软件的中间盒）。第二个重要研究领域是寻求将 SDN 概念从 AS 内部设置扩展到 AS 之间设置［Gupta 2014］。

实践原则

SDN 控制器学习案例：OpenDaylight 和 ONOS 控制器

在 SDN 发展早期，采用单一的 SDN 协议（OpenFlow［McKeown 2008；OpenFlow 2009］）和单一的 SDN 控制器（NOX［Gude 2008］）。此后，尤其是 SDN 控制器的数量有了很大增长［Kreutz 2015］。某些 SDN 控制器是公司特有的和专用的，例如 ONIX［Koponen 2010］、瞻博网络的 Contrail［Juniper Contrail 2016］以及谷歌用于其 B4 广域网的控制器［Jain 2013］。而更多控制器是开源的并以各种各样的编程语言实现［Erickson 2013］。最近，OpenDaylight 控制器［OpenDaylight Lithium 2016］和 ONOS 控制器［ONOS 2016］在产业界得到广泛支持。它们都是开源的，并且是与 Linux 基金会合作开发的。

OpenDaylight 控制器

图 5-17 呈现了 OpenDaylight Lithium SDN 控制器平台的简化视图［OpenDaylight Lithium 2016］。ODL 控制器组件的主要部分与我们在图 5-15 中给出的那些组件严格对应。**网络服务应用程序**用于决定数据平面转发和其他服务（如防火墙和负载均衡）如何在受控交换机中完成。与图 5-15 中规范的控制器不同，ODL 控制器具有**两个**接口，通过这两个接口，应用程序可以与原生的控制器服务通信以及彼此之间相互通信：外部应用程序使用 REST 请求－响应 API 与控制器模块通信，通信运行在 HTTP 上。内部应用程序经过服务抽象层（SAL）互相通信。至于控制器应用程序是在外部还是内部实现，这是由应用程序设计者选择决定的，图 5-17 中显示的特定应用程序配置只是作为一个例子。

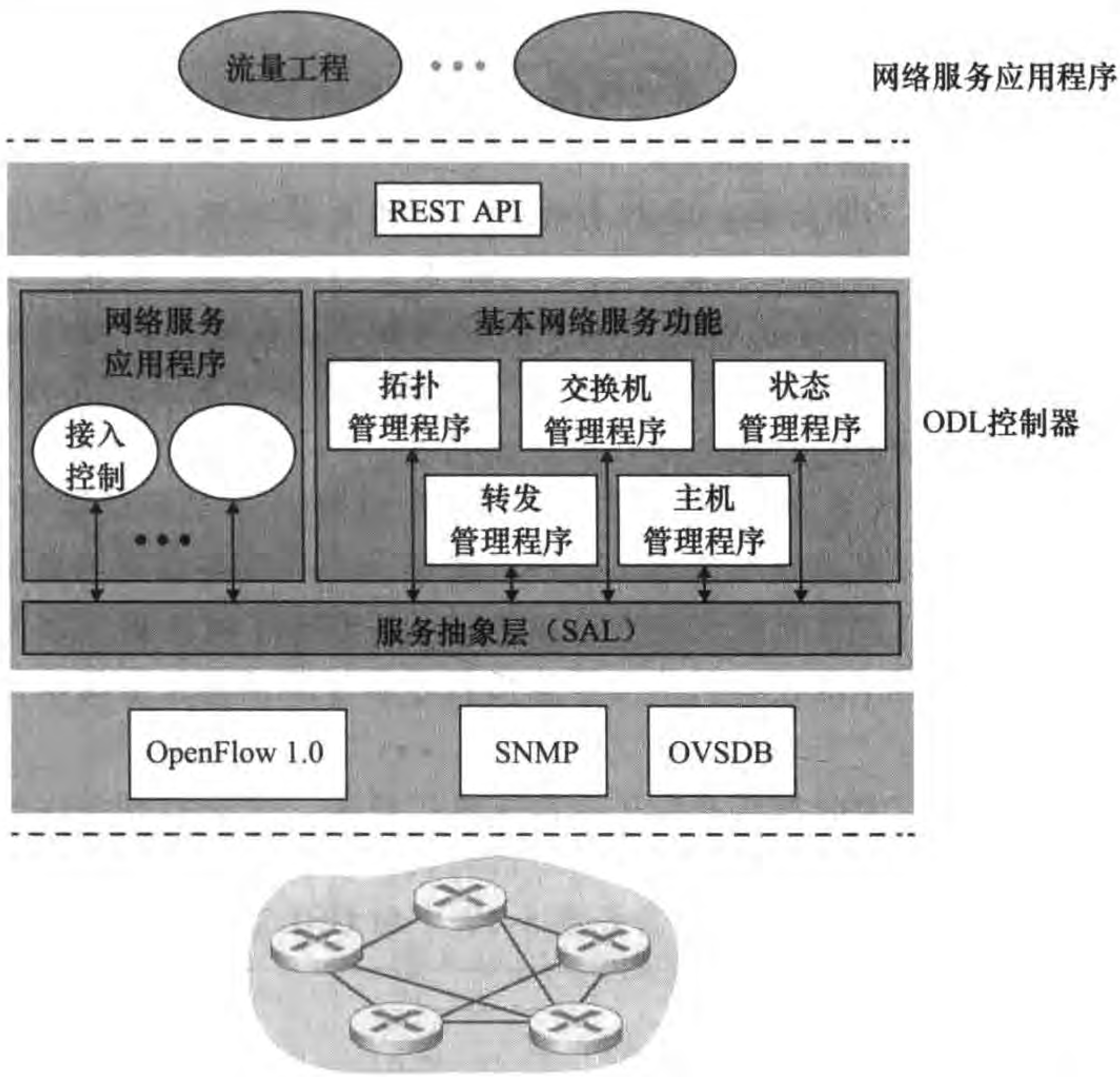


图 5-17　OpenDaylight 控制器

ODL 的**基本网络服务功能**是该控制器的核心，它们密切对应于我们在图 5-15 中遇到的网络范围状态管理能力。SAL 是控制器的神经中枢，允许控制器组件和应用程序互相调用服务并且订阅它们产生的事件。它也在通信层次对特定的**底层通信协议**提供了统一的抽象接口，包括 OpenFlow 和 SNMP（简单网络管理协议，即我们将在 5.7 节中涉及的一种网络管理协议）。OVSDB 是用于管理数据中心交换的协议，而数据中心交换是 SDN 技术的一个重要应用领域。我们将在第 6 章中介绍数据中心网络。

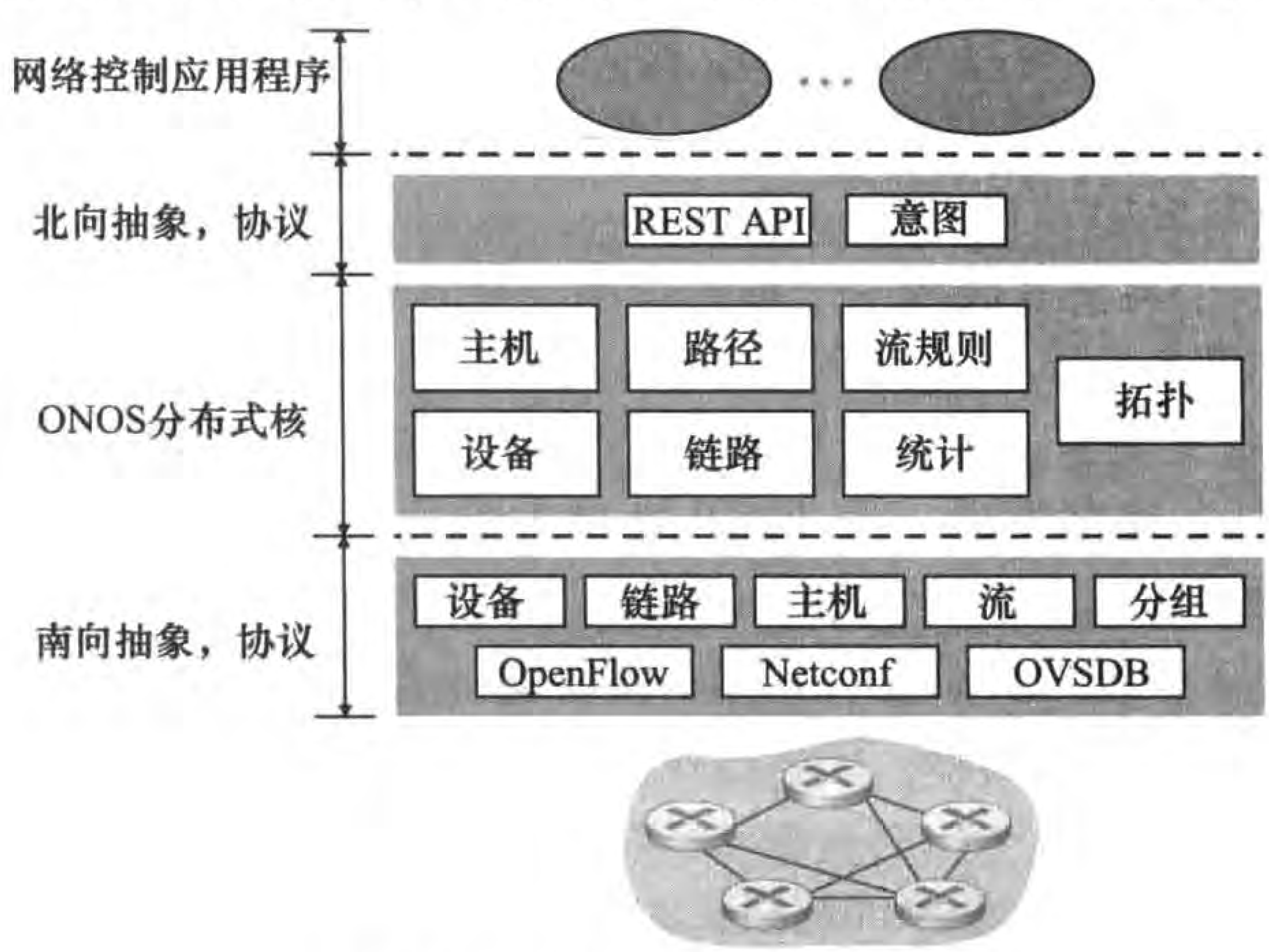


图 5-18　ONOS 控制器体系结构

ONOS 控制器

图 5-18 呈现了 ONOS 控制器的简化视图 [ONOS 2016]。类似于图 5-15 中的规范控制器，在 ONOS 控制器中有三个层次：

- **北向抽象和协议**。ONOS 的一个独有特征是它的意图框架，它允许应用程序请求高层服务（例如，在主机 A 和主机 B 之间建立一条连接，或相反地不允许主机 A 和主机 B 通信），而不必知道该服务的执行细节。或者以同步的方式（经过请求），或者以异步的方式（经过监听程序回调，例如等网络状态改变时），经过北向 API 向网络控制的应用程序提供状态信息。
- **分布式核**。ONOS 的分布式核中维护了网络的链路、主机和设备的状态。ONOS 被部署为在一系列互联的服务器上的一种服务，每台服务器运行着 ONOS 软件的相同副本，增加服务器数量就提升了服务能力。ONOS 核提供了在实例之间服务复制和协同的机制，这种机制为上层应用程序和下层网络设备提供了逻辑上集中的核服务抽象。
- **南向抽象和协议**。南向抽象屏蔽了底层主机、链路、交换机和协议的异构性，允许分布式核对设备和协议不可知。因为这种抽象，位于分布式核下方的南向接口逻辑上比图 5-14 中的规范控制器或图 5-17 中的 ODL 控制器更高。

5.6 ICMP：因特网控制报文协议

由 [RFC 792] 定义的因特网控制报文协议（ICMP），被主机和路由器用来彼此沟通网络层的信息。ICMP 最典型的用途是差错报告。例如，当运行一个 HTTP 会话时，你也许会遇到一些诸如“目的网络不可达”之类的错误报文。这种报文就来源于 ICMP。在某个位置，IP 路由器不能找到一条通往 HTTP 请求中所指定的主机的路径，该路由器就会向你的主机生成并发出一个 ICMP 报文以指示该错误。

ICMP 通常被认为是 IP 的一部分，但从体系结构上讲它位于 IP 之上，因为 ICMP 报文是承载在 IP 分组中的。这就是说，ICMP 报文是作为 IP 有效载荷承载的，就像 TCP 与 UDP 报文段作为 IP 有效载荷被承载那样。类似地，当一台主机收到一个指明上层协议为 ICMP 的 IP 数据报时（上层协议编码为 1），它分解出该数据报的内容给 ICMP，就像分解出一个数据报的内容给 TCP 或 UDP 一样。

ICMP 报文有一个类型字段和一个编码字段，并且包含引起该 ICMP 报文首次生成的 IP 数据报的首部和前 8 个字节（以便发送方能确定引发该差错的数据报）。在图 5-19 中显示了所选的 ICMP 报文类型。注意到 ICMP 报文并不仅是用于通知差错情况。

众所周知的 ping 程序发送一个 ICMP 类型 8 编码 0 的报文到指定主机。看到回显（echo）请求，目的主机发回一个类型 0 编码 0 的 ICMP 回显回答。大多数 TCP/IP 实现直接在操作系统中支持 ping 服务器，即该服务器不是一个进程。[Stevens 1990] 的第 11 章提供了有关 ping 客户程序的源码。注意到客户程序需要能够指示操作系统产生一个类型 8 编码 0 的 ICMP 报文。

另一个有趣的 ICMP 报文是源抑制报文。这种报文在实践中很少使用。其最初目的是执行拥塞控制，即使得拥塞的路由器向一台主机发送一个 ICMP 源抑制报文，以强制该主机减小其发送速率。我们在第 3 章已看到，TCP 在运输层有自己的拥塞控制机制，不需要

利用网络层中的反馈信息，如 ICMP 源抑制报文。

ICMP 类型	编码	描述
0	0	回显回答（对 ping 的回答）
3	0	目的网络不可达
3	1	目的主机不可达
3	2	目的协议不可达
3	3	目的端口不可达
3	6	目的网络未知
3	7	目的主机未知
4	0	源抑制（拥塞控制）
8	0	回显请求
9	0	路由器通告
10	0	路由器发现
11	0	TTL 过期
12	0	IP 首部损坏

图 5-19 ICMP 报文类型

在第 1 章中我们介绍了 Traceroute 程序，该程序允许我们跟踪从一台主机到世界上任意一台主机之间的路由。有趣的是，Traceroute 是用 ICMP 报文来实现的。为了判断源和目的地之间所有路由器的名字和地址，源主机中的 Traceroute 向目的地主机发送一系列普通的 IP 数据报。这些数据报的每个携带了一个具有不可达 UDP 端口号的 UDP 报文段。第一个数据报的 TTL 为 1，第二个的 TTL 为 2，第三个的 TTL 为 3，依次类推。该源主机也为每个数据报启动定时器。当第 n 个数据报到达第 n 台路由器时，第 n 台路由器观察到这个数据报的 TTL 正好过期。根据 IP 协议规则，路由器丢弃该数据报并发送一个 ICMP 告警报文给源主机（类型 11 编码 0）。该告警报文包含了路由器的名字和它的 IP 地址。当该 ICMP 报文返回源主机时，源主机从定时器得到往返时延，从 ICMP 报文中得到第 n 台路由器的名字与 IP 地址。

Traceroute 源主机是怎样知道何时停止发送 UDP 报文段的呢？前面讲过源主机为它发送的每个报文段的 TTL 字段加 1。因此，这些数据报之一将最终沿着这条路到达目的主机。因为该数据报包含了一个具有不可达端口号的 UDP 报文段，该目的主机将向源发送一个端口不可达的 ICMP 报文（类型 3 编码 3）。当源主机收到这个特别的 ICMP 报文时，知道它不需要再发送另外的探测分组。（标准的 Traceroute 程序实际上用相同的 TTL 发送 3 个一组的分组，因此 Traceroute 输出对每个 TTL 提供了 3 个结果。）

以这种方式，源主机知道了位于它与目的主机之间的路由器数量和标识，以及两台主机之间的往返时延。注意到 Traceroute 客户程序必须能够指令操作系统产生具有特定 TTL 值的 UDP 数据报，当 ICMP 报文到达时，也必须能够由它的操作系统进行通知。既然你已明白了 Traceroute 的工作原理，你也许想返回去更多地使用它。

在 RFC 4443 中为 IPv6 定义了 ICMP 的新版本。除了重新组织现有的 ICMP 类型和编码定义外，ICMPv6 还增加了新型 IPv6 功能所需的新类型和编码。这些包括“分组太大”类型和一个“未被承认的 IPv6 选项”差错编码。

5.7 网络管理和 SNMP

此时我们的网络层学习已经走到了结尾，我们前面仅有链路层了，我们都熟知网络是由许多复杂、交互的硬件和软件部件组成的，既包括构成网络的物理部件的链路、交换机、路由器、主机和其他设备，也包括控制和协调这些设备的许多协议。当一个机构将数以百计或数以千计的这种部件拼装在一起形成一个网络时，保持该网络"运行良好"对网络管理员无疑是一种挑战。我们在5.5节中看到，SDN环境中逻辑上集中的控制器能够有助于这种过程。但是网络管理的挑战在SDN出现前很久就已如影相随了，网络管理员使用丰富的网络管理工具和方法来监视、管理和控制该网络。在本节中我们将学习这些工具和技术。

一个经常被问到的问题是：什么是网络管理？我们用一个构思缜密的单句（虽然它相当冗长）来概括网络管理的定义［Saydam 1996］：

> "网络管理包括了硬件、软件和人类元素的设置、综合和协调，以监视、测试、轮询、配置、分析、评价和控制网络及网元资源，用合理的成本满足实时性、运营性能和服务质量的要求。"

给定了这个宽泛的定义，本节我们将仅涉及网络管理的入门知识，即网络管理员在执行其任务中所使用的体系结构、协议和信息库。我们将不涉及网络管理员的决策过程，其中故障标识［Labovitz 1997；Steinder 2002；Feamster 2005；Wu 2005；Teixeira 2006］、异常检测［Lakhina 2005；Barford 2009］、满足约定的服务等级约定（Service Level Agreements，SLA）的网络设计/工程［Huston 1999a］等主题会加以考虑。因此我们有意识地收窄关注点，有兴趣的读者应当参考这些文献：由Subramanian撰写的优秀的网络管理教科书［Subramanian 2000］，以及本书Web网站上可用的详尽的网络管理材料。

5.7.1 网络管理框架

图5-20显式了网络管理的关键组件。

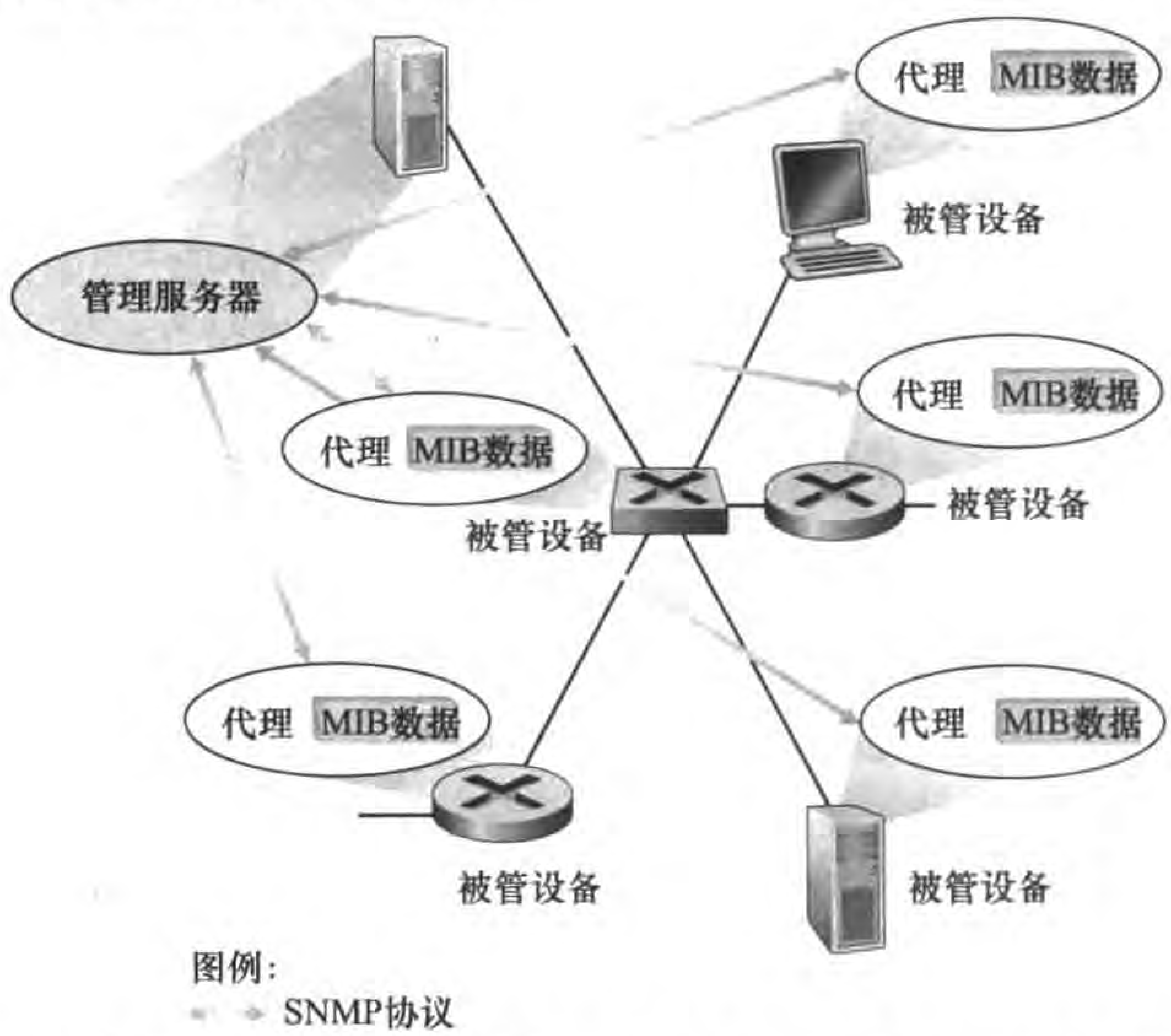


图5-20 网络管理的组件：管理服务器、被管设备、MIB数据，远程代理，SNMP

- **管理服务器**（managing server）是一个应用程序，通常有人的参与，并运行在网络运营中心（NOC）的集中式网络管理工作站上。管理服务器是执行网络管理活动的地方，它控制网络管理信息的收集、处理、分析和/或显示。正是在这里，发起控制网络行为的动作，人类网络管理员与网络设备打交道。
- **被管设备**（managed device）是网络装备的一部分（包括它的软件），位于被管理的网络中。被管设备可以是一台主机、路由器、交换机、中间盒、调制解调器、温度计或其他联网的设备。在一个被管设备中，有几个所谓**被管对象**（managed object）。这些被管对象是被管设备中硬件的实际部分（例如，一块网络接口卡只是一台主机或路由器的一个组件）和用于这些硬件及软件组件的配置参数（例如，像OSPF 这样的 AS 内部路由选择协议）。
- 一个被管设备中的每个被管对象的关联信息收集在**管理信息库**（Management Information Base，MIB）中，我们将看到这些信息的值可供管理服务器所用（并且在许多场合下能够被设置）。一个 MIB 对象可以是：一个计数器，例如由于 IP 数据报首部差错而由路由器丢弃的 IP 数据报的数量，或一台主机接收到的 UDP 报文段的数量；运行在一台 DNS 服务器上的软件版本的描述性信息；诸如一个特定设备功能是否正确的状态信息；或诸如到一个目的地的路由选择路径的特定协议的信息。MIB 对象由称为 SMI（Structure of Management Information）[RFC 2578；RFC 2579；RFC 2580] 的数据描述语言所定义。使用形式化定义语言可以确保网络管理数据的语法和语义是定义良好的和无二义性的。相关的 MIB 对象被收集在 MIB 模块（module）中。到 2015 年年中，RFC 定义了大约 400 个 MIB 模块，还有大量厂商特定的（专用的）MIB 模块。
- 在每个被管设备中还驻留有**网络管理代理**（network management agent），它是运行在被管设备中的一个进程，该进程与管理服务器通信，在管理服务器的命令和控制下在被管设备中采取本地动作。网络管理代理类似于我们在图 5-2 中看到的路由选择代理。
- 网络管理框架的最后组件是**网络管理协议**（network management protocol）。该协议运行在管理服务器和被管设备之间，允许管理服务器查询被管设备的状态，并经过其代理间接地在这些设备上采取行动。代理能够使用网络管理协议向管理服务器通知异常事件（如组件故障或超过了性能阈值）。重要的是注意到网络管理协议自己不能管理网络。恰恰相反，它为网络管理员提供了一种能力，使他们能够管理（“监视、测试、轮询、配置、分析、评价和控制”）网络。这是一种细微但却重要的区别。在下节中，我们将讨论因特网的 SNMP 协议。

5.7.2 简单网络管理协议

简单网络管理协议（Simple Network Management Protocol）版本 2（SNMPv2）[RFC 3416] 是一个应用层协议，用于在管理服务器和代表管理服务器执行的代理之间传递网络管理控制和信息报文。SNMP 最常使用的是请求响应模式，其中 SNMP 管理服务器向 SNMP 代理发送一个请求，代理接收到该请求后，执行某些动作，然后对该请求发送一个回答。请求通常用于查询（检索）或修改（设置）与某被管设备关联的 MIB 对象值。SNMP 第二个常被使用的是代理向管理服务器发送的一种非请求报文，该报文称为**陷阱报文**（trap message）。陷阱报文用于通知管理服务器，一个异常情况（例如一个链路接口启

动或关闭）已经导致了 MIB 对象值的改变。

表5-2中显示了 SNMPv2 定义的7种类型的报文，这些报文一般称为协议数据单元（PDU）。图5-21显示了这些 PDU 的格式。

表5-2 SNMPv2 PDU 类型

SNMPv2 PDU 类型	发送方 - 接收方	描述
GetRequest	管理者到代理	取得一个或多个 MIB 对象实例值
GetNextRequest	管理者到代理	取得列表或表格中下一个 MIB 对象实例值
GetBulkRequest	管理者到代理	以大数据块方式取得值，例如大表中的值
InformRequest	管理者到管理者	向不能访问的远程管理实体通知 MIB 值
SetRequest	管理者到代理	设置一个或多个 MIB 对象实例的值
Response	代理到管理者或管理者到管理者	对 GetRequest，GetNextRequest，GetBulkRequest，SetRequest PDU，或 InformRequest 产生的响应
SNMPv2 - Trap	代理到管理者	向管理者通知一个异常事件

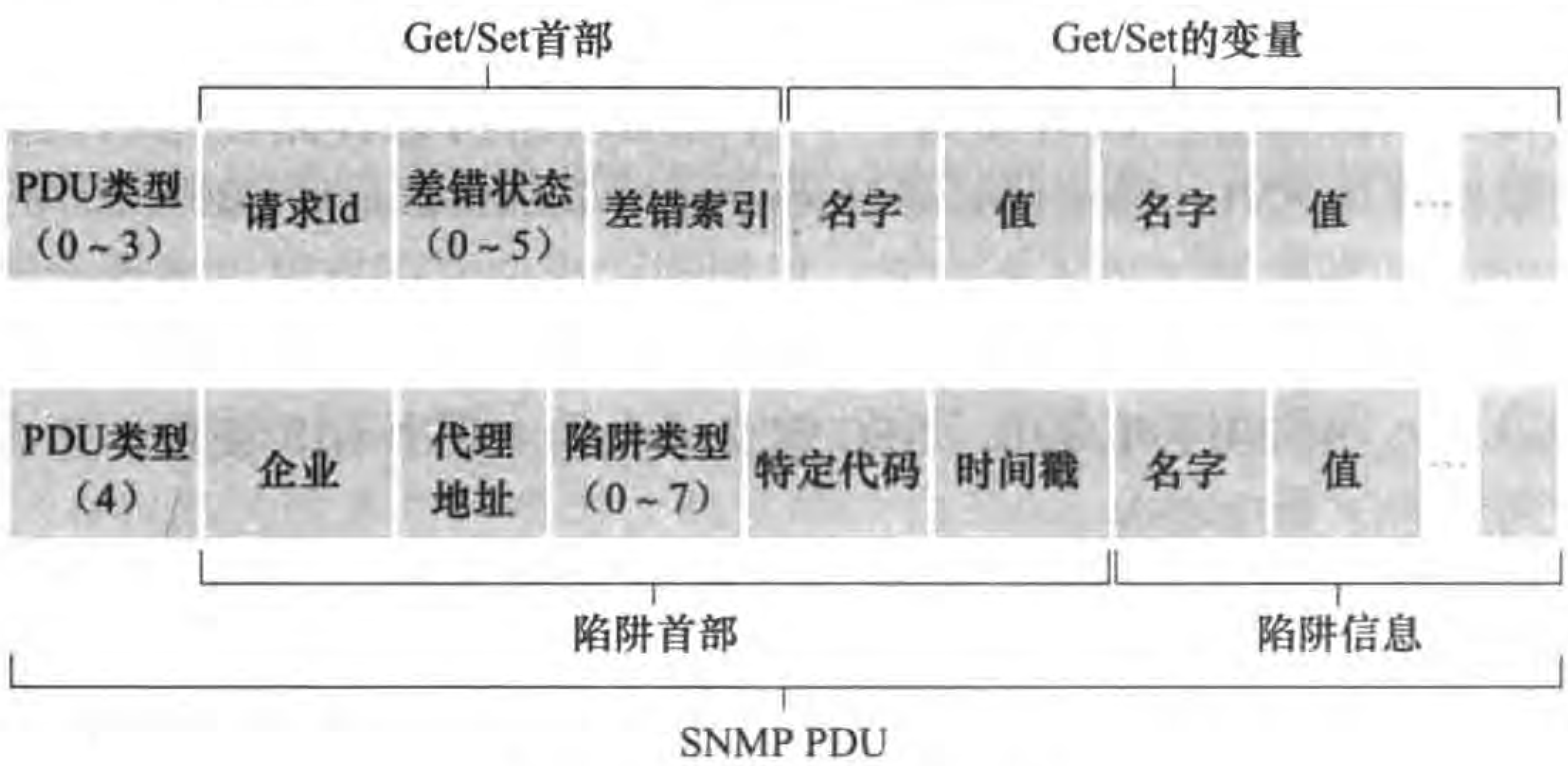


图5-21 SNMP PDU 格式

- GetRequest、GetNextRequest 和 GetBulkRequest PDU 都是管理服务器向代理发送的，以请求位于该代理所在的被管设备中的一个或多个 MIB 对象值。其值被请求的 MIB 对象的对象标识符定义在该 PDU 的变量绑定部分。GetRequest、GetNextRequest 和 GetBulkRequest 的差异在于它们的数据请求粒度。GetRequest 能够请求 MIB 值的任意集合；多个 GetNextRequest 能用于顺序地读取 MIB 对象的列表或表格；GetBulkRequest 允许读取大块数据，能够避免因发送多个 GetRequest 或 GetNextRequest 报文可能导致的额外开销。在所有这三种情况下，代理用包括该对象标识符和它们相关值的 Response PDU 进行响应。
- 管理服务器使用 SetRequest PDU 来设置位于被管设备中的一个或多个 MIB 对象的值。代理用带有“noError”差错状态的 Response PDU 进行应答，以证实该值的确已被设置。
- 管理服务器使用 InformRequest PDU 来通知另一个 MIB 信息管理服务器，后者对于接收服务器是远程的。
- Response PDU 通常从被管设备发送给管理服务器，以响应来自该服务器的请求报文，返回所请求的信息。

- SNMPv2 PDU 的最后一种类型是陷阱报文。陷阱报文是异步产生的，即它们不是为了响应接收到的请求而产生的，而是为了响应管理服务器要求通知的事件而产生的。RFC 3418 定义了周知的陷阱类型，其中包括设备的冷启动或热启动、链路就绪或故障、找不到相邻设备，或鉴别失效事件。接收到的陷阱请求不要求从管理服务器得到响应。

考虑到 SNMPv2 请求响应性质，这时需要注意到尽管 SNMP PDU 能够通过许多不同的运输协议传输，但 SNMP PDU 通常是作为 UDP 数据报的载荷进行传输的。RFC 3417 的确表明 UDP 是“首选的运输映射”。然而，由于 UDP 是一种不可靠的运输协议，因而不能确保一个请求或它的响应能够被它希望的目的地接收到。管理服务器用该 PDU 的请求 ID 字段为它向代理发送的请求编号；该代理的响应从接收到的请求中获取它的请求 ID。因此，该请求 ID 字段能被管理服务器用来检测丢失的请求或回答。如果在一定时间后还没有收到对应的响应，由管理服务器来决定是否重传一个请求。特别是，SNMP 标准没有强制任何特殊的重传过程，即使初始进行重传。它只是要求管理服务器“需要根据重传的频率和周期做出负责任的动作”。当然，这使人想知道一个“负责任的”协议应当如何动作！

SNMP 经历了 3 个版本的演变。SNMPv3 的设计者说过“SNMPv3 能被认为是具有附加安全性和管理能力的 SNMPv2”［RFC 3410］。SNMPv3 无疑在 SNMPv2 基础上有改变，而没有什么比在管理和安全领域的变化更为明显。在 SNMPv3 中，安全性的中心地位特别重要，因为缺乏适当的安全性导致 SNMP 主要用于监视而不是控制（例如，在 SNMPv1 中很少使用 SetRequest）。我们再一次看到安全性是重要的关注点（安全性是第 8 章详细学习的主题），尽管认识到它的重要性也许有些迟了，但“亡羊补牢，犹未为晚”。

5.8 小结

我们现在已经完成了进入网络核心的两章旅程，即开始于第 4 章的网络层数据平面的学习和本章完成的网络层控制平面的学习。我们知道了控制平面是网络范围的逻辑，它不仅控制从源主机到目的主机沿着端到端路径在路由器之间如何转发数据报，而且控制网络层组件和服务器如何配置和管理。

我们学习了构建控制平面有两大类方法：传统的*每路由器控制*（其中在每台路由器中运行算法，并且路由器中的路由选择组件与其他路由器中的路由选择组件通信）和*软件定义网络（SDN）控制*（其中一个逻辑上集中的控制器计算并向每台路由器分发转发表为它们所用）。我们在 5.2 节中学习了两种基本的路由选择算法，即链路状态和距离矢量，用于计算图中的最小开销路径；这些算法在每路由器控制和 SDN 控制中都有应用。这些算法是两种广泛部署的因特网路由选择协议 OSPF 和 BGP 的基础，我们在 5.3 节和 5.4 节中讨论了这两种协议。我们在 5.5 节中讨论了网络层控制平面的 SDN 方法，研究了 SDN 网络控制应用程序、SDN 控制器，以及控制器和 SDN 控制设备之间通信所使用的 OpenFlow 协议。在 5.6 节和 5.7 节中，我们包括了管理 IP 网络的某些技术细节：ICMP（互联网控制报文协议）和 SNMP（简单网络管理协议）。

在完成了网络层学习之后，我们的旅行此时沿着协议栈向下走了一步，即到了链路层。像网络层一样，链路层是每台网络连接的设备的一部分。但我们将在下一章中看到，链路层的任务是在相同链路或局域网之间更局域化地移动分组。尽管这种任务从表面上看

可能比网络层任务简单得多，但我们将看到，链路层涉及许多重要和引人入胜的问题，这些问题会花费我们不少时间。

课后习题和问题

复习题

5.1 节

R1. 基于每路由器控制的控制平面意味着什么？在这种情况下，当我们说网络控制平面和数据平面是“整体地”实现时，是什么意思？

R2. 基于逻辑上集中控制的控制平面意味着什么？在这种有情况下，数据平面和控制平面是在相同的设备或在分离的设备中实现的吗？请解释。

5.2 节

R3. 比较和对照集中式和分布式路由选择算法的性质。给出一个路由选择协议的例子，该路由选择协议采用分布式方法和集中式方法。

R4. 比较和对照链路状态和距离矢量这两种路由选择算法。

R5. 在距离矢量路由选择中的“无穷计数”是什么意思？

R6. 每个自治系统使用相同的 AS 内部路由选择算法是必要的吗？说明其原因。

5.3~5.4 节

R7. 为什么在因特网中用到了不同的 AS 间与 AS 内部协议？

R8. 是非判断题：当一台 OSPF 路由器发送它的链路状态信息时，它仅向那些直接相邻的节点发送。解释理由。

R9. 在 OSPF 自治系统中区域表示什么？为什么引入区域的概念？

R10. 定义和对比下列术语：子网、前缀和 BGP 路由。

R11. BGP 是怎样使用 NEXT-HOP 属性的？它是怎样使用 AS-PATH 属性的？

R12. 描述一个较高层 ISP 的网络管理员在配置 BGP 时是如何实现策略的。

R13. 是非判断题：当 BGP 路由器从它的邻居接收到一条通告的路径时，它必须对接收路径增加上它自己的标识，然后向其所有邻居发送该新路径。解释理由。

5.5 节

R14. 描述在 SDN 控制器中的通信层、网络范围状态管理层和网络控制应用程序层的主要任务。

R15. 假定你要在 SDN 控制平面中实现一个新型路由选择协议。你将在哪个层次中实现该协议？解释理由。

R16. 什么类型的报文流跨越 SDN 控制器的北向和南向 API？谁是从控制器跨越南向接口发送的这些报文的接收者？谁是跨越北向接口从控制器发送的这些报文的接收者？

R17. 描述两种从受控设备到控制器发送的 OpenFlow 报文（由你所选）类型的目的。描述两种从控制器到受控设备发送的 OpenFlow 报文（由你所选）类型的目的。

R18. 在 OpenDaylight SDN 控制器中服务抽象层的目的是什么？

5.6~5.7 节

R19. 列举 4 种不同类型的 ICMP 报文。

R20. 在发送主机执行 Traceroute 程序，收到哪两种类型的 ICMP 报文？

R21. 在 SNMP 环境中定义下列术语：管理服务器、被管设备、网络管理代理和 MIB。

R22. SNMP GetRequest 和 SetRequest 报文的目的是什么？

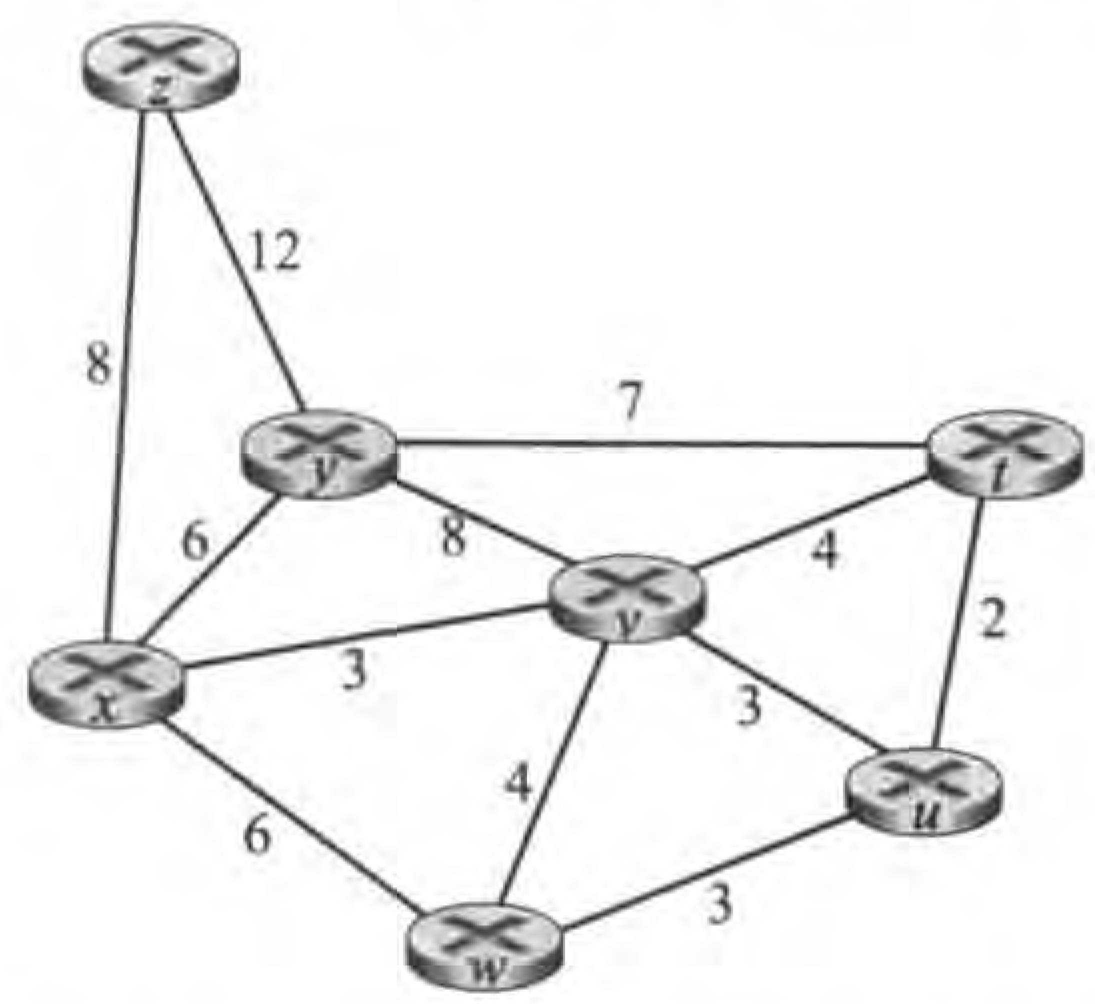
R23. SNMP 陷阱报文的目的是什么？

习题

P1. 观察图 5-3，列举从 y 到 u 不包含任何环路的路径。

P2. 重复习题 P1，列举从 x 到 z、z 到 u 以及 z 到 w 的不包含任何环路的路径。

P3. 考虑下面的网络。对于标明的链路开销，用 Dijkstra 的最短路算法计算出从 x 到所有网络节点的最短路径。通过计算一个类似于表 5-1 的表，说明该算法是如何工作的。



P4. 考虑习题 P3 中所示的网络。使用 Dijkstra 算法和一个类似于表 5-1 的表来说明你做的工作：

a. 计算出从 t 到所有网络节点的最短路径。

b. 计算出从 u 到所有网络节点的最短路径。

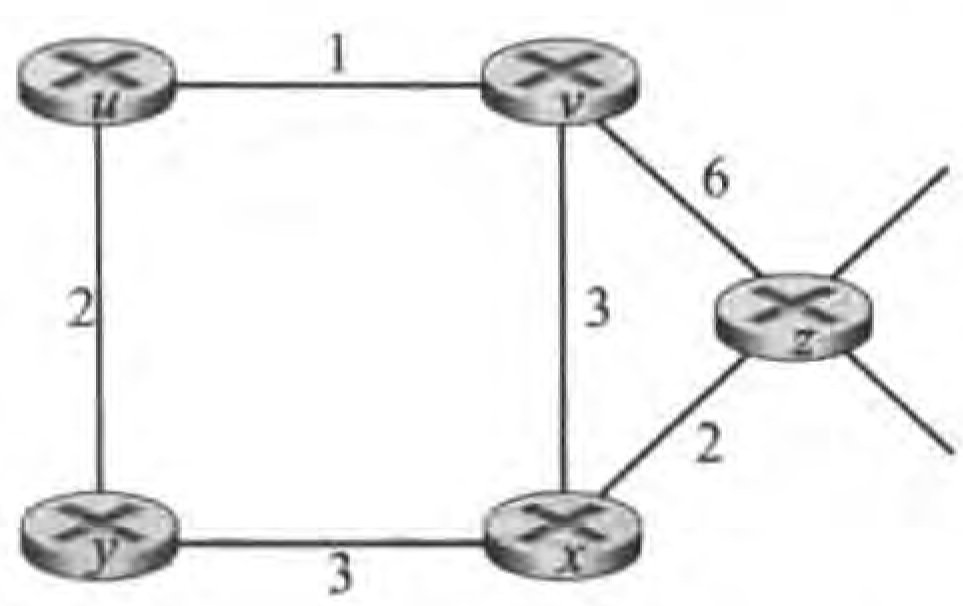
c. 计算出从 v 到所有网络节点的最短路径。

d. 计算出从 w 到所有网络节点的最短路径。

e. 计算出从 y 到所有网络节点的最短路径。

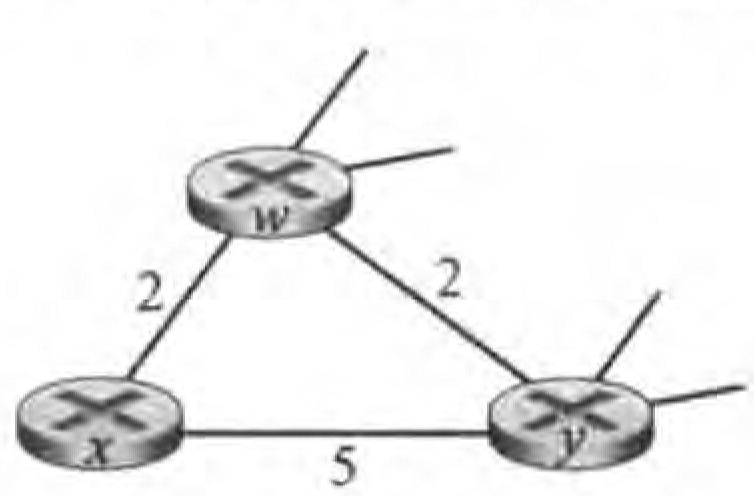
f. 计算出从 z 到所有网络节点的最短路径。

P5. 考虑下图所示的网络，假设每个节点初始时知道到它的每个邻居的开销。考虑距离向量算法，并显示在节点 z 中的距离表表项。



P6. 考虑一个一般性拓扑（即不是以上所显示的特定网络）和一个同步版本的距离向量算法。假设每次迭代时，一个节点与其邻居交换其距离向量并接收它们的距离向量。假定算法开始时，每个节点只知道到其直接邻居的开销，在该分布式算法收敛前所需的最大迭代次数是多少？评估你的答案。

P7. 考虑下图所示的网络段。x 只有两个相连邻居 w 与 y。w 有一条通向目的地 u（没有显示）的最低开销路径，其值为 5，y 有一条通向目的地 u 的最低开销路径，其值为 6。从 w 与 y 到 u（以及 w 与 y 之间）的完整路径未显示出来。网络中所有链路开销皆为正整数值。



a. 给出 x 对目的地 w、y 和 u 的距离向量。

b. 给出对于 $c(x, w)$ 或 $c(x, y)$ 的链路开销的变化，使得执行了距离向量算法后，x 将通知其邻居有一条通向 u 的新最低开销路径。

c. 给出对 $c(x, w)$ 或 $c(x, y)$ 的链路开销的变化，使得执行了距离向量算法后，x 将不通知其邻居有一条通向 x 的新最低开销路径。

P8. 考虑如图5-6中所示3个节点的拓扑。不使用显示在图5-6中的开销值，链路开销值现在是 $c(x, y)=3$，$c(y, z)=6$，$c(z, x)=4$。在距离向量表初始化后和在同步版本的距离向量算法每次迭代后，计算它的距离向量表（如我们以前对图5-6讨论时所做的那样）。

P9. 考虑距离向量路由选择中的无穷计数问题。如果我们减小一条链路的开销，将会出现无穷计数问题吗？为什么？如果我们将没有链路的两个节点连接起来，会出现什么情况？

P10. 讨论图5-6中的距离向量算法，距离向量 $\boldsymbol{D}(x)$ 中的每个值不是递增的并且最终将在有限步中稳定下来。

P11. 考虑图5-7。假定有另一台路由器 w，与路由器 y 和 z 连接。所有链路的开销给定如下：$c(x, y)=4$，$c(x, z)=50$，$c(y, w)=1$，$c(z, w)=1$，$c(y, z)=3$。假设在距离向量路由选择算法中使用了毒性逆转。

a. 当距离向量路由选择稳定时，路由器 w、y 和 z 向 x 通知它们之间的距离。它们告诉彼此什么样的距离值？

b. 现在假设 x 和 y 之间的链路开销增加到60。如果使用了毒性逆转，将会存在无穷计数问题吗？为什么？如果存在无穷计数问题，距离向量路由选择需要多少次迭代才能再次到达稳定状态？评估你的答案。

c. 如果 $c(y, x)$ 从4变化到60，怎样修改 $c(y, z)$ 使得不存在无穷计数问题。

P12. 描述在BGP中是如何检测路径中的环路的。

P13. BGP路由器将总是选择具有最短AS路径长度的无环路由吗？评估你的答案。

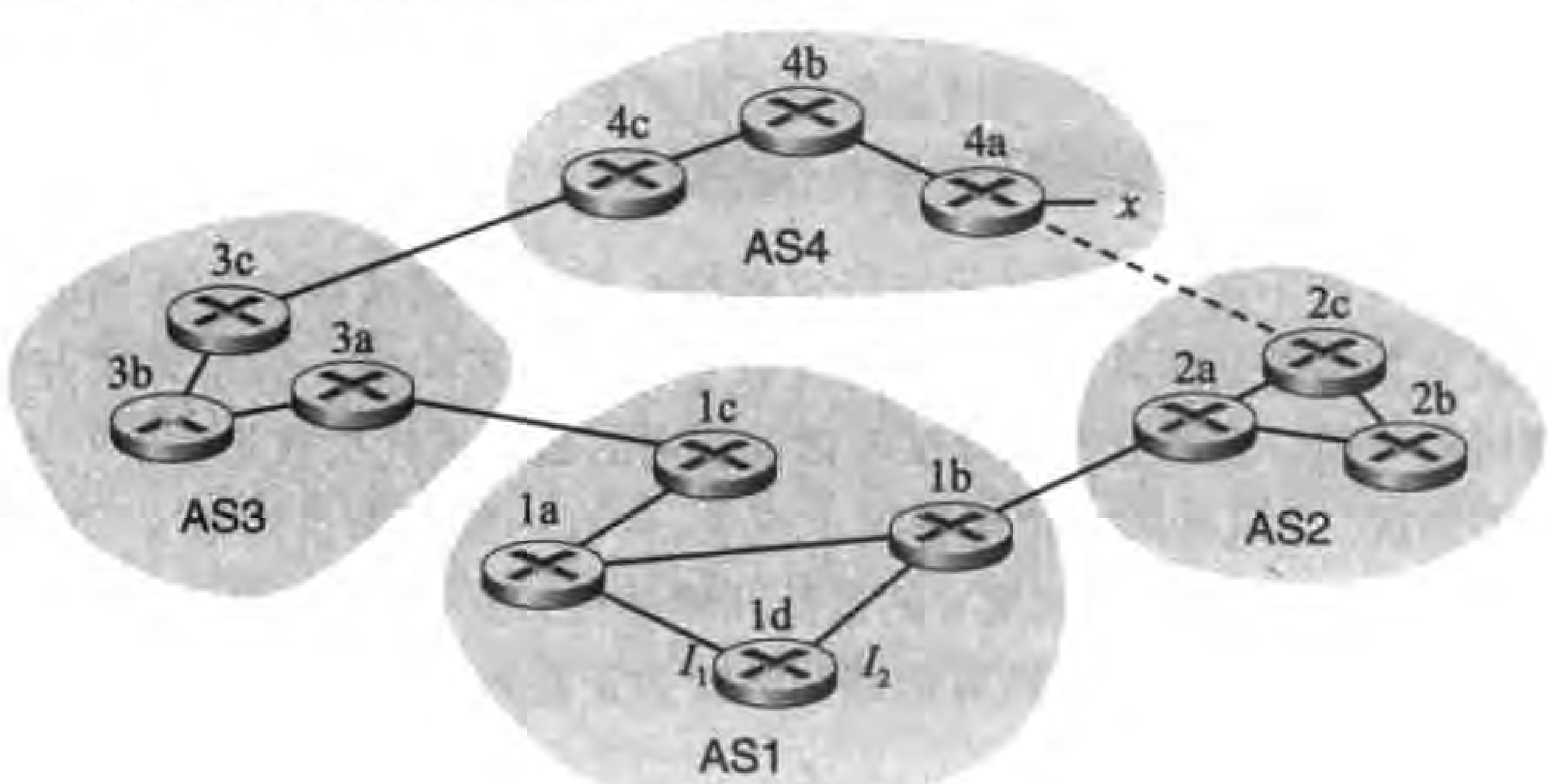
P14. 考虑下图所示的网络。假定AS3和AS2正在运行OSPF作为其AS内部路由选择协议。假定AS1和AS4正在运行RIP作为其AS内部路由选择协议。假定AS间路由选择协议使用的是eBGP和iBGP。假定最初在AS2和AS4之间不存在物理链路。

a. 路由器3c从下列哪个路由选择协议学习到了前缀 x：OSPF、RIP、eBGP或iBGP？

b. 路由器3a从哪个路由选择协议学习到了前缀 x？

c. 路由器1c从哪个路由选择协议学习到了前缀 x？

d. 路由器1d从哪个路由选择协议学习到了前缀 x？



P15. 参考前面习题P14，一旦路由器1d知道了 x 的情况，它将一个表项 (x, I) 放入它的转发表中。

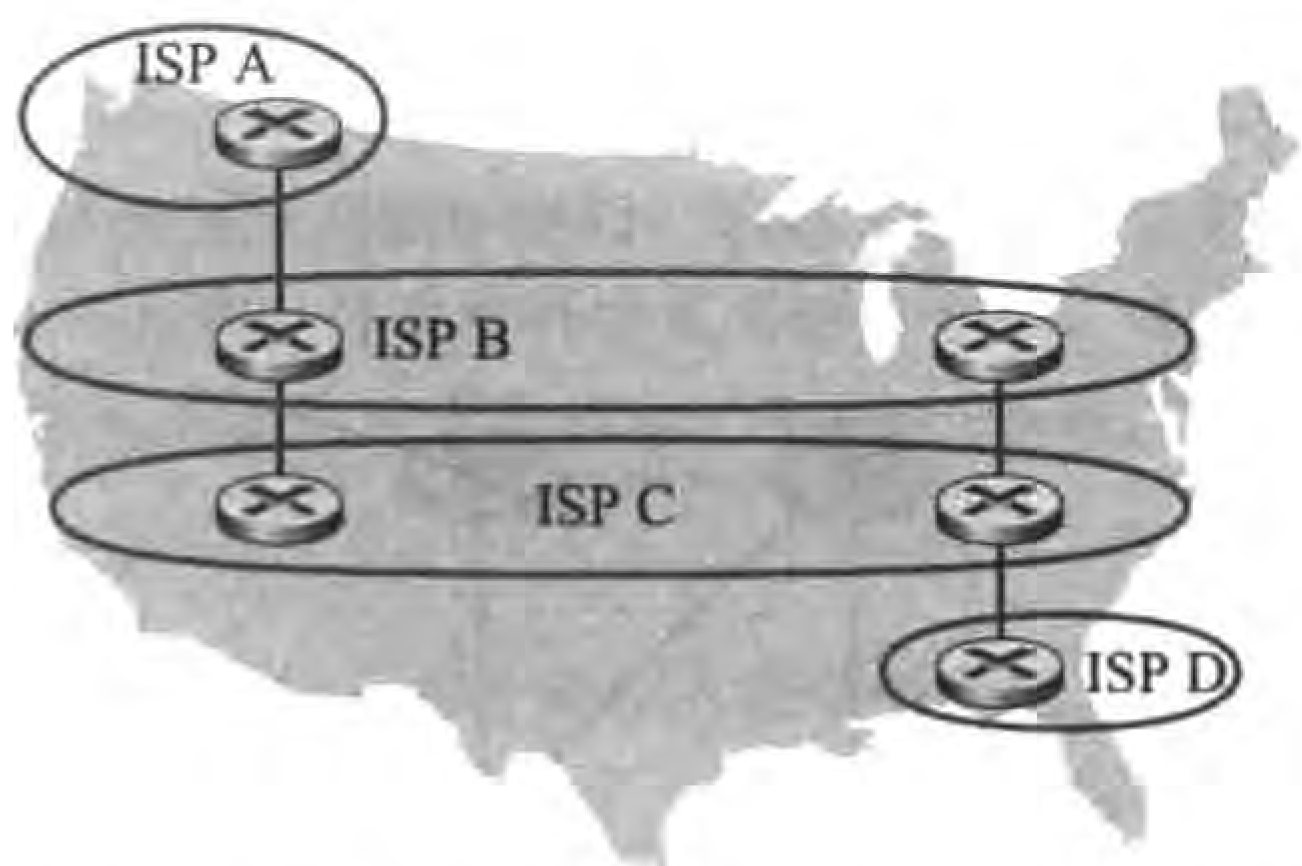
a. 对这个表项而言，I 将等于 I_1 还是 I_2？用一句话解释其原因。

b. 现在假定在AS2和AS4之间有一条物理链路，显示为图中的虚线。假定路由器1d知道经AS2以及经AS3能够访问到 x。I 将设置为 I_1 还是 I_2？用一句话解释其原因。

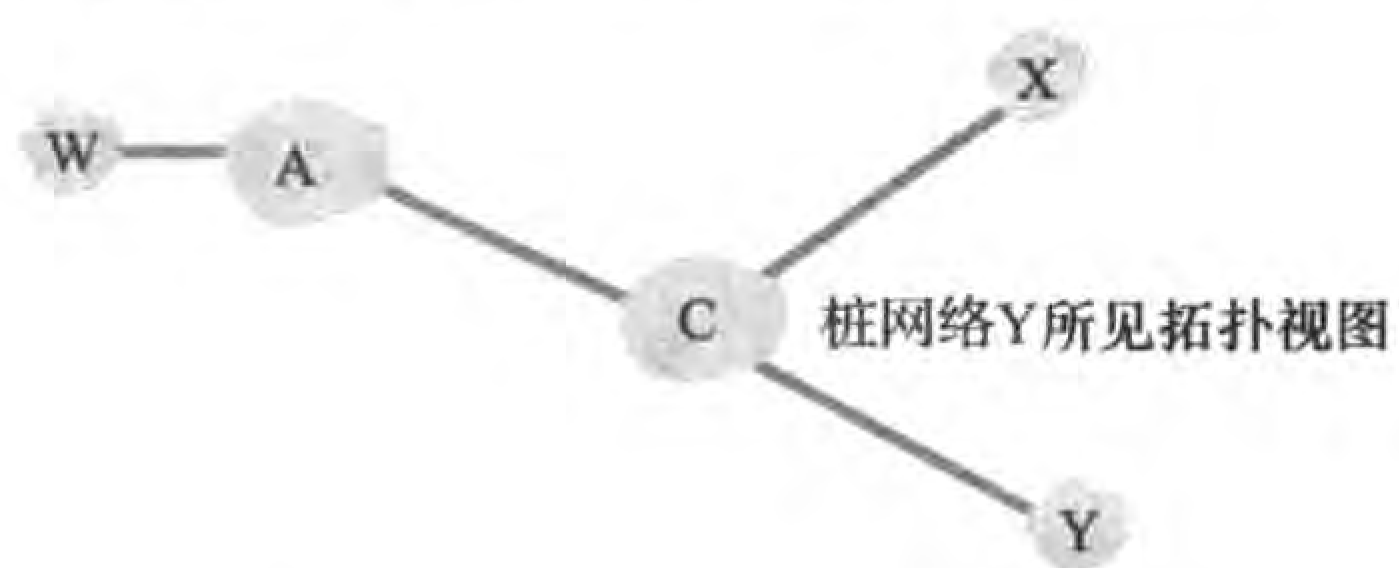
c. 现在假定有另一个AS，它称为AS5，其位于路径AS2和AS4之间（没有显示在图中）。假定路由

器 1d 知道经 AS2 AS5 AS4 以及经过 AS3 AS4 能够访问到 x。I 将设置为 I_1 还是 I_2？用一句话解释其原因。

P16. 考虑下面的网络。ISP B 为地区 ISP A 提供国家级主干服务。ISP C 为地区 ISP D 提供国家级主干服务。每个 ISP 由一个 AS 组成。B 和 C 使用 BGP，在两个地方互相对等。考虑从 A 到 D 的流量。B 愿意将流量交给 C 传给西海岸（使得 C 将承担承载跨越整个国家的流量开销），而 C 愿意经其东海岸与 B 对等的站点得到这些流量（使得 B 将承载跨越整个国家的流量）。C 可能会使用什么样的 BGP 机制，使得 B 将通过东海岸对等点传递 A 到 D 的流量？要回答这个问题，你需要钻研 BGP 规范。



P17. 在图 5-13 中，考虑到达桩网络 W、X 和 Y 的路径信息。基于 W 与 X 处的可用信息，它们分别看到的网络拓扑是什么？评估你的答案。Y 所见的拓扑视图如下图所示。



P18. 考虑图 5-13。B 将不会基于 BGP 路由选择，经过 X 以 Y 为目的地转发流量。但是有某些极为流行的应用程序，其数据分组先朝向 X，然后再流向 Y。指出一种这样的应用程序，描述数据分组是如何沿着这条未由 BGP 路由选择所给定的路径流动的。

P19. 在图 5-13 中，假定有另一个桩网络 V，它为 ISP A 的客户。假设 B 和 C 具有对等关系，并且 A 是 B 和 C 的客户。假设 A 希望让发向 W 的流量仅来自 B，并且发向 V 的流量来自 B 或 C。A 如何向 B 和 C 通告其路由？C 收到什么样的 AS 路由？

P20. 假定 AS X 和 Z 不直接连接，但与 AS Y 连接。进一步假定 X 与 Y 具有对等协定，Y 与 Z 具有对等协定。最后，假定 Z 要传送所有 Y 的流量但不传送 X 的流量。BGP 允许 Z 实现这种策略吗？

P21. 考虑在管理实体和被管设备之间发生通信的两种方式：请求响应方式和陷阱方式。从以下方面考虑这两种方式的优缺点：①开销；②当异常事件出现时通知的时间；③对于管理实体和设备之间丢失报文的健壮性。

P22. 在 5.7 节中我们看到，用不可靠的 UDP 数据报传输 SNMP 报文是更可取的方式。请考虑 SNMP 设计者选择 UDP 而不是 TCP 作为 SNMP 运输协议的理由。

套接字编程作业

在第 2 章结尾给出了 4 个套接字编程作业。下面给出第 5 个应用 ICMP 的作业（ICMP 的是本章讨论的协议）。

作业 5：ICMP ping

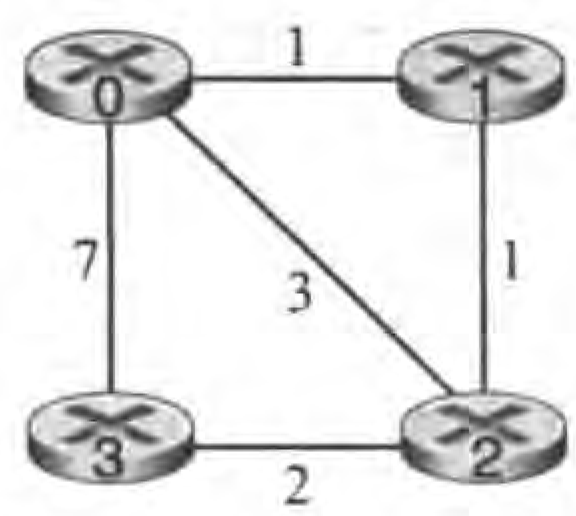
ping 是一种流行的网络应用程序，用于测试位于远程的某个特定的主机是否开机和可达。它也经常

用于测量客户主机和目标主机之间的时延。它的工作过程是：向目标主机发送 ICMP“回显请求”分组（即 ping 分组），并且侦听 ICMP“回显响应”应答（即 pong 分组）。ping 测量 RRT、记录分组丢失和计算多个 ping-pong 交换（往返时间的最小、平均、最大和标准差）的统计汇总。

在本实验中，读者将用 Python 语言编写自己的 ping 应用程序。该应用程序将使用 ICMP。但为了保持程序的简单，将不完全遵循 RFC 1739 中的官方规范。注意到仅需要写该程序的客户程序，因为服务器侧所需的功能构建在几乎所有的操作系统中。读者能够在 Web 站点 http://www.pearsonhighered.com/cs-resources 找到本作业的全面细节，以及该 Python 代码的重要片段。

编程作业

在本编程作业中，需要写一组“分布式”程序，以为下图所示的网络实现一个分布式异步距离向量路由选择算法。



要写出下列例程，这些例程将在为该作业提供的模拟环境中异步“执行”。对于节点 0，将要写出这样的例程：

- rtinit0()。在模拟开始将调用一次该例程。rtinit0()无参数。它应当初始化节点 0 中的距离表，以反映出到达节点 1、2 和 3 的直接开销分别为 1、3 和 7。在上图中，所有链路都是双向的，两个方向的开销皆相同。在初始化距离表和节点 0 的例程所需的其他数据结构后，它应向其直接连接的邻居（在本情况中为节点 1、2 和 3）发送它到所有其他网络节点的最低开销路径的开销信息。通过调用例程 tolayer2()，这种最低开销信息在一个路由选择更新分组中被发送给相邻节点，就像在完整编程作业中描述的那样。路由选择更新分组的格式也在完整编程作业中进行描述。
- rtupdate0(struct rtpkt *rcvdpkt)。当节点 0 收到一个由其直接相连邻居之一发给它的路由选择分组时，调用该例程。参数 *rcvdpkt 是一个指向接收分组的指针。rtupdate0()是距离向量算法的“核心”。它从其他节点 i 接收的路由选择更新分组中包含节点 i 到所有其他网络节点的当前最短路径开销值。rtupdate0()使用这些收到的值来更新其自身的距离表（这是由距离向量算法所规定的）。如果它自己到另外节点的最低开销由于此更新而发生改变的话，则节点 0 通过发送一个路由选择分组来通知其直接相连邻居这种最低开销的变化。我们在距离向量算法中讲过，仅有直接相连的节点才交换路由选择分组。因此，节点 1 和节点 2 将相互通信，但节点 1 和节点 3 将不相互通信。

为节点 1、2、3 定义类似的例程。因此你总共将写出 8 个例程：rtinit0()、rtinit1()、rtinit2()、rtinit3()、rtupdate0()、rtupdate1()、rtupdate2()和 rtupdate3()。这些例程将共同实现一个分布式的、与图中所示拓扑和开销相关的距离表的异步计算。

读者可在网址 http://www.pearsonhighered.com/cs-resources 处找到该编程作业的全部详细资料，以及创建模拟硬件/软件环境所需的 C 程序代码。一个 Java 版的编程作业也可供使用。

Wireshark 实验

在本书配套的 Web 站点 www.pearsonhighered.com/cs-resource 上，将找到一个 Wireshark 实验作业，该作业考察了在 ping 和 traceroute 命令中 ICMP 协议的使用。

人物专访



Jennifer Rexford

Jennifer Rexford是美国普林斯顿大学计算机科学系的教授。她的研究有宏大的目标：使计算机网络更容易设计和管理，特别强调路由选择协议。1996～2004年，她在AT&T实验室网络管理和性能部工作。在AT&T期间，她设计了用于网络测量、流量工程和路由器配置的技术和工具，这些技术和工具部署在AT&T的主干网络。Jennifer是《Web协议和实践：网络协议、高速缓存和流量测量》（Web Protocols and Practice：Networking Protocols，Caching，and Traffic Measurement）一书的合作者，该书由Addison-Wesley出版社于2001年5月出版。她在2003～2007年担任ACM SIGCOMM主席一职。于1991年从普林斯顿大学电气工程获得学士学位，并于1996年从密西根大学电气工程和计算机科学获得博士学位。在2004年，Jennifer赢得了ACM的计算机专业杰出青年Grace Murray Hopper奖，并入列35岁以下顶级创新家的MIT TR-100名单。

- 请描述在您的职业生涯中干过的一两个最令人激动的项目。工作中最大的挑战是什么？

当我是AT&T的研究员时，我们一群人设计了一种管理因特网服务提供商主干网络路由的新方法。传统上，网络操作员逐个配置每台路由器，并且这些路由器运行分布式协议以计算通过该网络的路径。我们认为如果网络操作员能够基于网络范围拓扑和流量的视图直接控制路由器如何转发流量，网络管理将更为简单和灵活。我们设计并建造的路由控制平台（RCP）能够在一台商用计算机上为所有AT&T主干计算路径，并且能够不加修改地控制老式路由器。对我而言，这个项目是令人兴奋的，因为我们有一个刺激性的想法、一个工作的系统以及最终在一个运营的网络中实际部署。快速向前走了几年，软件定义网络（SDN）已经成为一种主流技术，标准协议（如OpenFlow）可以告诉底层的交换机做些什么更为容易。

- 您认为软件定义网络未来应当如何演进？

与以往最大的不同之处在于，控制平面软件能够由许多不同的程序员创建，而不只是由销售网络设备的公司生成。同时不像在服务器或智能手机上运行的应用程序，控制器应用程序必须在一起工作以处理相同的流量。网络操作员不希望对某些流量执行负载均衡而对其他流量进行路由选择；相反，他们希望在相同的流量上执行负载均衡和路由选择。未来的SDN控制器平台应当使独立编写的多个控制器应用程序在一起合作，提供良好的编程抽象。更一般地，良好的编程抽象能够使得生成控制器应用程序更为容易，而不必担心诸如流表项、流量计数器、分组首部的比特样式等底层细节。此外，虽然SDN控制器是逻辑上集中的，但网络仍然是由分布式设备集合组成的。未来的控制器应当为跨网络更新流表提供良好抽象，因此应用程序能够在设备更新时推断出传输中的分组发生了什么情况。对控制平面软件的编程抽象是一个令人兴奋的计算机网络、分布式系统和编程语言之间的多学科研究领域，有望在未来几年产生实际影响。

- 您预见网络和因特网的未来往何处发展？

网络是一个令人兴奋的领域，因为应用程序和底层技术无时不在变化。我们总是在重塑自己！甚至在10年前，有谁能够预测到智能手机的一统天下，允许移动用户访问现有应用程序以及新型基于位置的服务呢？云计算的出现从根本上改变了用户与他们运行的应用程序之间的关系，联网的传感器和执行器（物联网）使得大量的新应用（和安全脆弱性）成为可能！创新的步伐真正令人兴奋不已。

底层网络是所有这些创新中的要素。固然，该网络声名狼藉地“造成不便”：限制了性能，损害了可靠性，约束了应用以及使服务的部署和管理复杂化。我们应当继续努力使得未来的网络就像我们呼吸的空气一样不可见，因此网络决不会成为新思想和有价值服务的拦路虎。为此，我们需要在各个网络设备和协议（以及它们的首字母缩略词）之上提升抽象等级，使得我们能够对该网络以及用户的高层目标作为一个整体进行推理。

- 是谁激发了您的职业灵感？

我在国际计算机科学研究所长期受到 Sally Floyd 的激励。她的研究总是有明确目标，聚焦因特网面临的重要挑战。她深入到困难问题之中，直到她完全理解该问题和解空间，她将大量精力专注于“使得事情产生结果”，例如在协议标准化和网络设备中注入了她的很多思想。同时，通过在许多标准化和研究组织的专业服务以及通过创建工具（例如广泛使用的 ns2 和 ns3 模拟器），她回馈了网络界，这些工具使得其他研究人员取得成功。她于 2009 年退休，但她在该领域的影响将在未来许多年内长久存在。

- 您对进入计算机科学和网络领域的学生有什么忠告吗？

网络本质上是一个跨学科的领域。应用来自其他学科的技术在网络中取得重要突破，这些技术来自不同领域，如排队论、博弈论、控制论、分布式系统、网络优化、编程语言、机器学习、算法、数据结构等等。我认为熟悉相关领域或与这些领域的专家密切合作，是将网络建立在更坚实基础上的极好方式，这样我们能够学习如何建造更值得社会信任的网络。除了这些理论学科以外，因为我们能够创造真实的供人们使用的实际人造物品，网络是令人激动的。通过获取操作系统、计算机体系结构等方面的经验，掌握如何设计和建造系统是另一种了不起的方式，这可以增强你的网络知识，有助于使得世界更美好。

链路层和局域网

在上一章中，我们学习了网络层提供的任意两台主机之间的通信服务。在两台主机之间，数据报跨越一系列通信链路传输，一些是有线链路，而一些是无线链路，从源主机起始，通过一系列分组交换机（交换机和路由器），在目的主机结束。当我们沿协议栈继续往下，从网络层到达链路层，我们自然而然地想知道分组是如何通过构成端到端通信路径的各段链路的。为了在单段链路上传输，网络层的数据报是怎样被封装进链路层帧的呢？沿此通信路径，不同的链路能够采用不同的链路层协议吗？在广播链路中传输碰撞是如何解决的？在链路层存在编址吗？如果需要，链路层编址如何与我们在第4章中学习的网络层编址一起运行呢？交换机和路由器之间到底有哪些差异？我们将在本章回答这些和其他一些重要的问题。

在链路层的讨论中，我们将看到两种截然不同类型的链路层信道。第一种类型是广播信道，这种信道用于连接有线局域网、卫星网和混合光纤同轴电缆（Hybrid Fiber Coaxial cable，HFC）接入网中的多台主机。因为许多主机与相同的广播信道连接，需要所谓的媒体访问协议来协调帧传输。在某些场合中，可以使用中心控制器来协调传输。第二种类型的链路层信道是点对点通信链路，这在诸如长距离链路连接的两台路由器之间，或用户办公室计算机与它们所连接的邻近以太网交换机之间等场合经常能够发现。协调对点对点链路的访问较为简单；在本书 Web 网站上的相关材料详细地讨论了点到点协议（Point-to-Point Protocol，PPP），该协议的适用范围从经电话线的拨号服务到经光纤链路的高速点到点帧传输。

我们将在本章中探究几个链路层概念和技术。我们将更深入地研究差错检测和纠正，这个主题我们在第3章中简要讨论过。我们将考虑多路访问网络和交换局域网，包括以太网，这是目前最流行的有线局域网技术。我们还将学习虚拟局域网和数据中心网络。尽管 WiFi 及更一般的无线局域网都属于链路层范围，但我们将在第7章才学习这些重要的主题。

6.1 链路层概述

我们首先学习一些有用的术语。在本章中为方便讨论，将运行链路层协议（即第2层）协议的任何设备均称为**节点**（node）。节点包括主机、路由器、交换机和 WiFi 接入点（在第7章中讨论）。我们也把沿着通信路径连接相邻节点的通信信道称为**链路**（link）。为了将一个数据报从源主机传输到目的主机，数据报必须通过沿端到端路径上的**各段链路**传输。举例来说，显示在图6-1下部的公司网络中，考虑从无线主机之一向服务器之一发送一个数据报。该数据报将实际通过6段链路：发送主机与 WiFi 接入点之间的 WiFi 链路，接入点和链路层交换机之间的以太网链路，链路层交换机与路由器之间的链路，两台路由器之间的链路，最后是交换机和服务器之间的以太网链路。在通过特定的链路时，传输节点将数据报封装在**链路层帧**中，并将该帧传送到链路中。

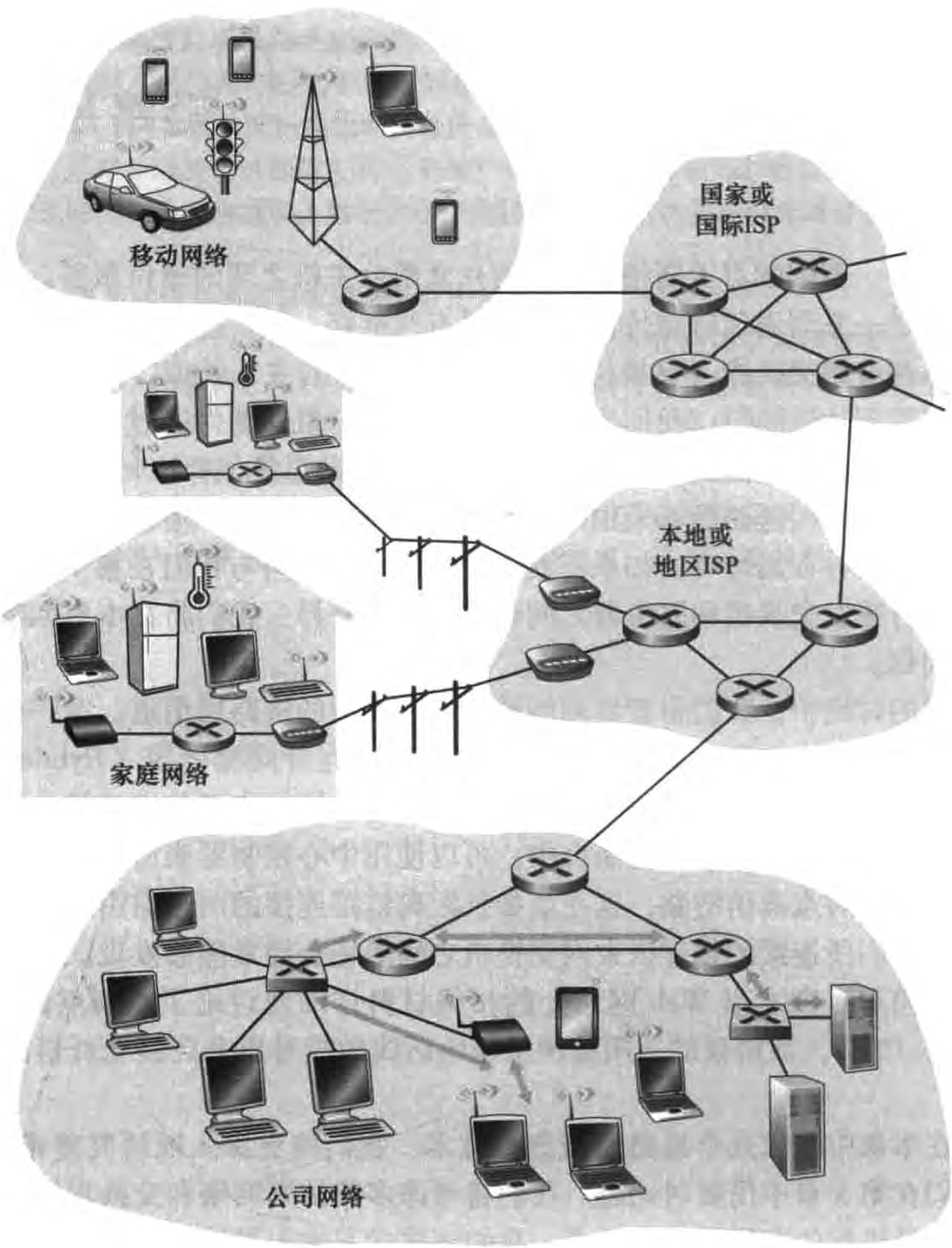


图6-1 无线主机和服务器之间的6个链路层跳

为了透彻理解链路层以及它是如何与网络层关联的，我们考虑一个交通运输的类比例子。假如一个旅行社计划为游客开辟从美国新泽西州的普林斯顿到瑞士洛桑的旅游路线。假定该旅行社认为对于游客而言最为便利的方案是：从普林斯顿乘豪华大轿车到JFK机场，然后乘飞机从JFK机场去日内瓦机场，最后乘火车从日内瓦机场到洛桑火车站。一旦该旅行社作了这3项预定，普林斯顿豪华大轿车公司将负责将游客从普林斯顿带到JFK，航空公司将负责将游客从JFK带到日内瓦，瑞士火车服务将负责将游客从日内瓦带到洛桑。该旅程中3段中的每一段都在两个“相邻”地点之间是“直达的”。注意到这3段运输是由不同的公司管理，使用了完全不同的运输方式（豪华大轿车、飞机和火车）。尽管运输方式不同，但它们都提供了将旅客从一个地点运输到相邻地点的基本服务。在这个运输类比中，一个游客好比一个数据报，每个运输区段好比一条链路，每种运输方式好比一种链路层协议，而该旅行社好比一个路由选择协议。

6.1.1 链路层提供的服务

尽管任一链路层的基本服务都是将数据报通过单一通信链路从一个节点移动到相邻节点，但所提供的服务细节能够随着链路层协议的不同而变化。链路层协议能够提供的可能服务包括：

- 成帧（framing）。在每个网络层数据报经链路传送之前，几乎所有的链路层协议都要将其用链路层帧封装起来。一个帧由一个数据字段和若干首部字段组成，其中网络层数据报就插在数据字段中。帧的结构由链路层协议规定。当我们在本章的后半部分研究具体的链路层协议时，将看到几种不同的帧格式。
- 链路接入。**媒体访问控制**（Medium Access Control，MAC）协议规定了帧在链路上传输的规则。对于在链路的一端仅有一个发送方、链路的另一端仅有一个接收方的点对点链路，MAC 协议比较简单（或者不存在），即无论何时链路空闲，发送方都能够发送帧。更有趣的情况是当多个节点共享单个广播链路时，即所谓多路访问问题。这里，MAC 协议用于协调多个节点的帧传输。
- 可靠交付。当链路层协议提供可靠交付服务时，它保证无差错地经链路层移动每个网络层数据报。前面讲过，某些运输层协议（例如 TCP）也提供可靠交付服务。与运输层可靠交付服务类似，链路层的可靠交付服务通常是通过确认和重传取得的（参见 3.4 节）。链路层可靠交付服务通常用于易于产生高差错率的链路，例如无线链路，其目的是本地（也就是在差错发生的链路上）纠正一个差错，而不是通过运输层或应用层协议迫使进行端到端的数据重传。然而，对于低比特差错的链路，包括光纤、同轴电缆和许多双绞铜线链路，链路层可靠交付可能会被认为是一种不必要的开销。由于这个原因，许多有线的链路层协议不提供可靠交付服务。
- 差错检测和纠正。当帧中的一个比特作为 1 传输时，接收方节点中的链路层硬件可能不正确地将其判断为 0，反之亦然。这种比特差错是由信号衰减和电磁噪声导致的。因为没有必要转发一个有差错的数据报，所以许多链路层协议提供一种机制来检测这样的比特差错。通过让发送节点在帧中包括差错检测比特，让接收节点进行差错检查，以此来完成这项工作。第 3 章和第 4 章讲过，因特网的运输层和网络层也提供了有限形式的差错检测，即因特网检验和。链路层的差错检测通常更复杂，并且用硬件实现。差错纠正类似于差错检测，区别在于接收方不仅能检测帧中出现的比特差错，而且能够准确地确定帧中的差错出现的位置（并因此纠正这些差错）。

6.1.2 链路层在何处实现

在深入学习链路层的细节之前，本概述的最后一节考虑一下在何处实现链路层的问题。我们将关注一个端系统，因为我们在第 4 章中知道链路层是实现在路由器的线路卡中的。主机的链路层是用硬件还是用软件实现的呢？它是实现在一块单独的卡上还是一个芯片上？它是怎样与主机的硬件和操作系统组件的其他部分接口的呢？

图 6-2 显示了一个典型的主机体系结构。链路层的主体部分是在**网络适配器**（network adapter）中实现的，网络适配器有时也称为**网络接口卡**（Network Interface Card，NIC）。位于网络适配器核心的是链路层控制器，该控制器通常是一个实现了许多链路层服务（成帧、链路接入、差错检测等）的专用芯片。因此，链路层控制器的许多功能是用硬件实现

的。例如，Intel 的 710 控制器［Intel 2016］实现了以太网协议，我们将在 6.5 节中学习该协议；Atheros AR5006［Atheros 2016］控制器实现了 802.11 WiFi 协议，我们将在第 7 章学习该协议。直到 20 世纪 90 年代后期，大部分网络适配器还是物理上分离的卡（如一块 PCMCIA 卡或者一块插进 PC 的 PCI 卡槽中的插入卡），但越来越多的网络适配器被综合进主机的主板，即所谓的局域网在主板配置。

在发送端，控制器取得了由协议栈较高层生成并存储在主机内存中的数据报，在链路层帧中封装该数据报（填写该帧的各个字段），然后遵循链路接入协议将该帧传进通信链路中。在接收端，控制器接收了整个帧，抽取出网络层数据报。如果链路层执行差错检测，则需要发送控制器在该帧的首部设置差错检测比特，由接收控制器执行差错检测。

图 6-2 显示了与主机总线（例如一条 PCI 或 PCI-X 总线）连接的网络适配器，这里它看起来非常像与其他主机组件连接的任何其他 I/O 设备。图 6-2 还显示了尽管大部分链路层是在硬件中实现的，但部分链路层是在运行于主机 CPU 上的软件中实现的。链路层的软件组件实现了高层链路层功能，如组装链路层寻址信息和激活控制器硬件。在接收端，链路层软件响应控制器中断（例如，由于一个或多个帧的到达），处理差错条件和将数据报向上传递给网络层。所以，链路层是硬件和软件的结合体，即此处是协议栈中软件与硬件交接的地方。［Intel 2016］从软件编程的角度提供了有关 XL 710 控制器的可读性很强的概述（以及详细的描述）。

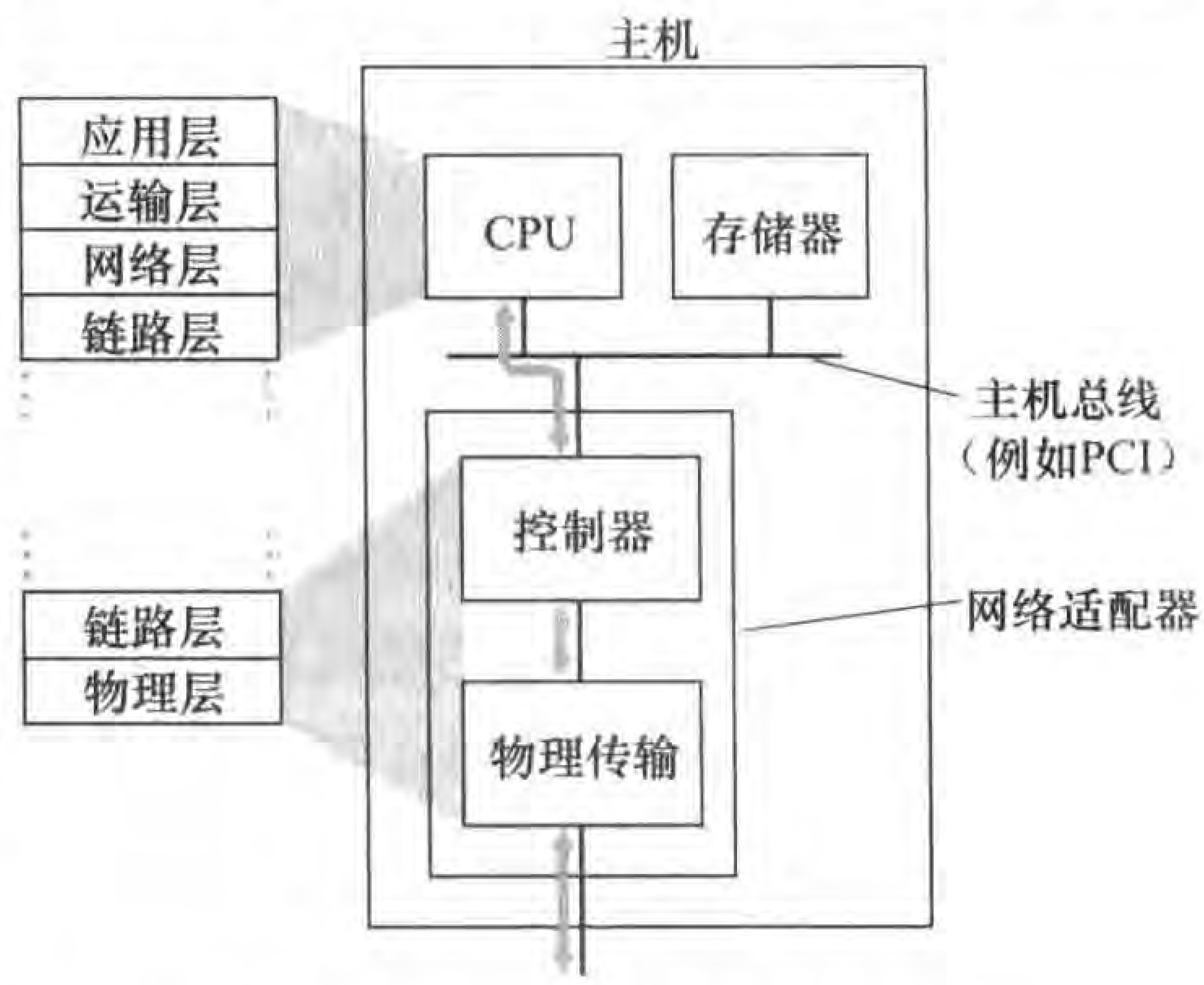


图 6-2 网络适配器：它与其他主机组件及协议栈功能的关系

6.2 差错检测和纠正技术

在上一节中，我们提到了**比特级差错检测和纠正**（bit-level error detection and correction），即对从一个节点发送到另一个物理上连接的邻近节点的链路层帧中的比特损伤进行检测和纠正，它们通常是链路层提供的两种服务。我们在第 3 章中看到差错检测和纠正服务通常也由运输层提供。在本节中，我们将研究几种最简单的技术，它们能够用于检测比特差错，而且在某些情况下，能够纠正这样的比特差错。对该主题理论和实现的全面描述是许多教科书的主题（例如［Schwartz 1980］或［Bertsekas 1991］），而我们这里仅讨论必要内容。我们此时的目的是对差错检测和纠正技术提供的能力有一种直观的认识，并看看一些简单技术在链路层中的工作原理及其如何实际应用。

图 6-3 图示说明了我们研究的环境。在发送节点，为了保护比特免受差错，使用**差错检测和纠正比特**（Error-Detection and-Correction，EDC）来增强数据 *D*。通常，要保护的数据不仅包括从网络层传递下来需要通过链路传输的数据报，而且包括链路帧首部中的链路级的寻址信息、序号和其他字段。链路级帧中的 *D* 和 *EDC* 都被发送到接收节点。在接收节点，接收到比特序列 *D'* 和 *EDC'*。注意到因传输中的比特翻转所致，*D'* 和 *EDC'* 可能与初始的 *D* 和 *EDC* 不同。

接收方的挑战是在它只收到 *D'* 和 *EDC'* 的情况下，确定 *D'* 是否和初始的 *D* 相同。在

图6-3中的接收方判定的准确措辞（我们问是否检测到一个差错，而非是否出现了差错！）是重要的。差错检测和纠正技术使接收方有时但并总是检测出已经出现的比特差错。即使采用差错检测比特，也还是可能有**未检出比特差错**（undetected bit error）；这就是说，接收方可能无法知道接收的信息中包含着比特差错。因此，接收方可能向网路层交付一个损伤的数据报，或者不知道该帧首部的某个其他字段的内容已经损伤。我们因此要选择一个差错检测方案，使得这种事件发生的概率很小。一般而言，差错检测和纠错技术越复杂（即那些具有未检测出比特差错概率较小的技术），导致的开销就越大，这就是意味着需要更多的计算量及更多的差错检测和纠错比特。

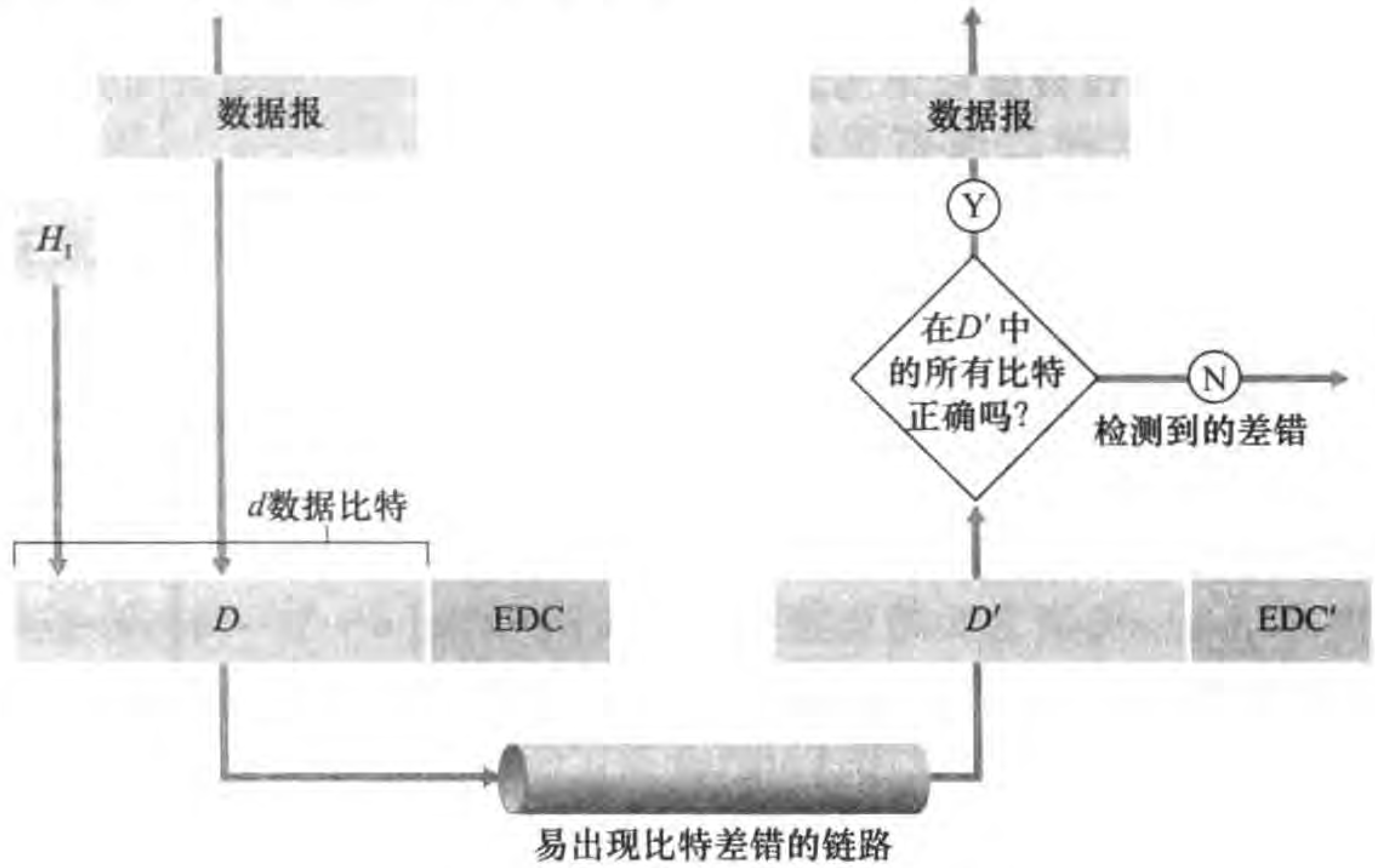


图6-3 差错检测与纠正的场景

我们现在来研究在传输数据中检测差错的3种技术：奇偶校验（它用来描述差错检测和纠正背后隐含的基本思想）、检验和方法（它通常更多地应用于运输层）和循环冗余检测（它通常更多地应用在适配器中的链路层）。

6.2.1 奇偶校验

也许差错检测最简单的方式就是用单个**奇偶校验位**（parity bit）。假设在图6-4中要发送的信息D有d比特。在偶校验方案中，发送方只需包含一个附加的比特，选择它的值，使得这d+1比特（初始信息加上一个校验比特）中1的总数是偶数。对于奇校验方案，选择校验比特值使得有奇数个1。图6-4描述了一个偶校验的方案，单个校验比特被存放在一个单独的字段中。

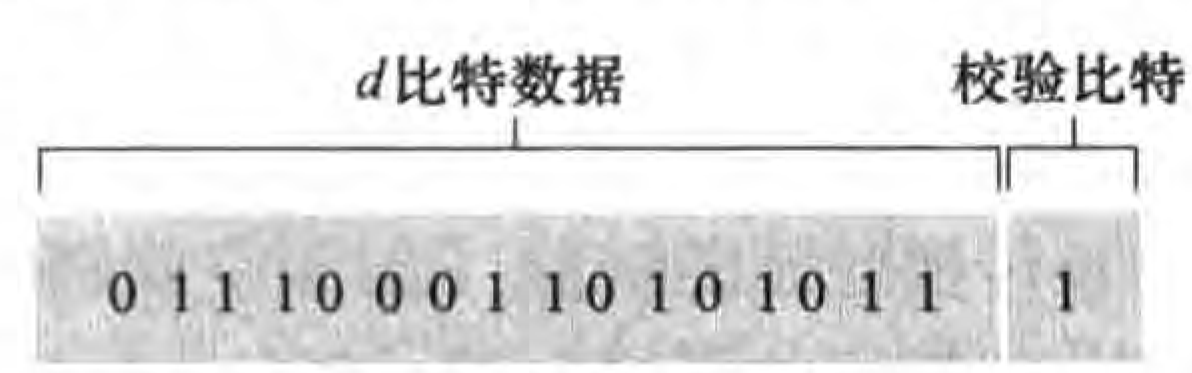


图6-4 1比特偶校验

采用单个奇偶校验位方式，接收方的操作也很简单。接收方只需要数一数接收的d+1比特中1的数目即可。如果在采用偶校验方案中发现了奇数个值为1的比特，接收方知道至少出现了一个比特差错。更精确的说法是，出现了*奇数个比特差错*。

但是如果出现了偶数个比特差错，那会发生什么现象呢？你应该认识到这将导致一个未检出的差错。如果比特差错的概率小，而且比特之间的差错可以被看作是独立发生的，在一个分组中多个比特同时出错的概率将是极小的。在这种情况下，单个奇偶校验位可能

是足够的了。然而，测量已经表明了差错经常以“突发”方式聚集在一起，而不是独立地发生。在突发差错的情况下，使用单比特奇偶校验保护的一帧中未检测出差错的概率能够达到50%［Spragins 1991］。显然，需要一个更健壮的差错检测方案（幸运的是实践中正在使用这样的方式!）。但是在研究实践中使用的差错检测方案之前，我们考虑对单比特奇偶校验的一种简单一般化方案，这将使我们深入地理解纠错技术。

图6-5显示了单比特奇偶校验方案的二维一般化方案。这里 D 中的 d 个比特被划分为 i 行 j 列。对每行和每列计算奇偶值。产生的 $i+j+1$ 奇偶比特构成了链路层帧的差错检测比特。

现在假设在初始 d 比特信息中出现了单个比特差错。使用这种**二维奇偶校验**（two-dimensional parity）方案，包含比特值改变的列和行的校验值都将会出现差错。因此接收方不仅可以检测到出现了单个比特差错的事实，而且还可以利用存在奇偶校验差错的列和行的索引来实际识别发生差错的比特并纠正它！图6-5显示了一个例子，其中位于（2，2）的值为1的比特损坏了，变成了0，该差错就是一个在接收方可检测并可纠正的差错。尽管我们的讨论是针对初始 d 比特信息的，但校验比特本身的单个比特差错也是可检测和可纠正的。二维奇偶校验也能够检测（但不能纠正!）一个分组中两个比特差错的任何组合。二维奇偶校验方案的其他特性将在本章后面的习题中进行探讨。

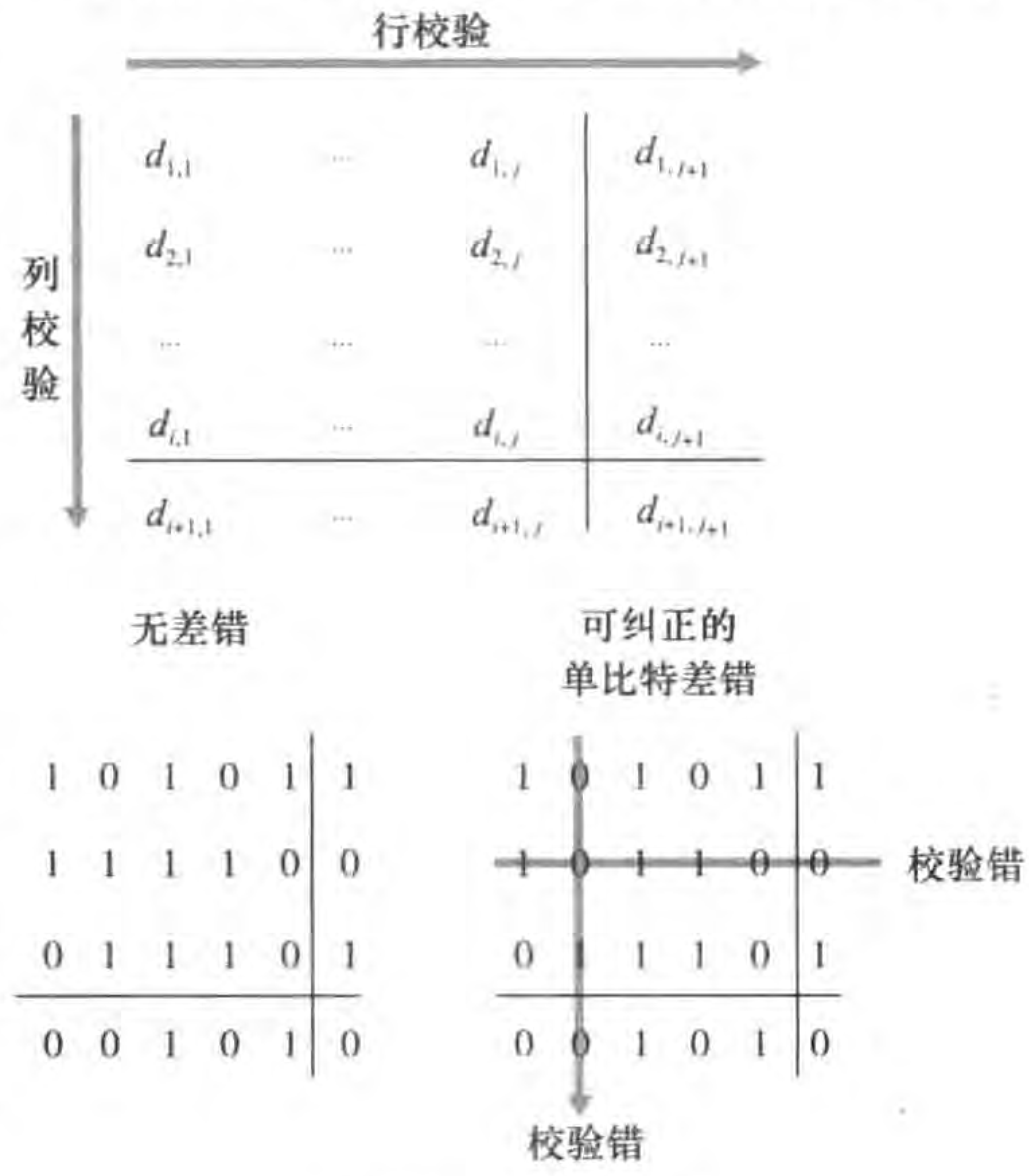


图6-5　二维偶校验

接收方检测和纠正差错的能力被称为**前向纠错**（Forward Error Correction，FEC）。这些技术通常用于如音频CD这样的音频存储和回放设备中。在网络环境中，FEC技术可以单独应用，或与链路层ARQ技术一起应用，ARQ技术与我们在第3章研究的协议类似。FEC技术很有价值，因为它们可以减少所需的发送方重发的次数。也许更为重要的是，它们允许在接收方立即纠正差错。FEC避免了不得不等待的往返时延，而这些时延是发送方收到NAK分组并向接收方重传分组所需要的，这对于实时网络应用［Rubenstein 1998］或者具有长传播时延的链路（如深空间链路）可能是一种非常重要的优点。研究差错控制协议中FEC的使用的资料包括［Biersack 1992；Nonnenmacher 1998；Byers 1998；Shacham 1990］。

6.2.2　检验和方法

在检验和技术中，图6-4中的 d 比特数据被作为一个 k 比特整数的序列处理。一个简单检验和方法就是将这 k 比特整数加起来，并且用得到的和作为差错检测比特。**因特网检验和**（Internet checksum）就基于这种方法，即数据的字节作为16比特的整数对待并求和。这个和的反码形成了携带在报文段首部的因特网检验和。如在3.3节讨论的那样，接收方通过对接收的数据（包括检验和）的和取反码，并且检测其结果是否为全1比特来检测检验和。如果这些比特中有任何比特是0，就可以指示出差错。RFC 1071详细地讨论因

特网检验和算法和它的实现。在 TCP 和 UDP 协议中，对所有字段（包括首部和数据字段）都计算因特网检验和。在其他协议中，例如 XTP [Strayer 1992]，对首部计算一个检验和，对整个分组计算另一个检验和。

检验和方法需要相对小的分组开销。例如，TCP 和 UDP 中的检验和只用了 16 比特。然而，与后面要讨论的常用于链路层的 CRC 相比，它们提供相对弱的差错保护。这时，一个很自然的问题是：为什么运输层使用检验和而链路层使用 CRC 呢？前面讲过运输层通常是在主机中作为用户操作系统的一部分用软件实现的。因为运输层差错检测用软件实现，采用简单而快速如检验和这样的差错检测方案是重要的。在另一方面，链路层的差错检测在适配器中用专用的硬件实现，它能够快速执行更复杂的 CRC 操作。Feldmeier[Feldmeier 1995] 描述的快速软件实现技术不仅可用于加权检验和编码，而且可用于 CRC（见后面）和其他编码。

6.2.3 循环冗余检测

现今的计算机网络中广泛应用的差错检测技术基于**循环冗余检测**（Cyclic Redundancy Check，CRC）**编码**。CRC 编码也称为**多项式编码**（polynomial code），因为该编码能够将要发送的比特串看作为系数是 0 和 1 一个多项式，对比特串的操作被解释为多项式算术。

CRC 编码操作如下。考虑 d 比特的数据 D，发送节点要将它发送给接收节点。发送方和接收方首先必须协商一个 $r+1$ 比特模式，称为**生成多项式**（generator），我们将其表示为 G。我们将要求 G 的最高有效位的比特（最左边）是 1。CRC 编码的关键思想如图 6-6 所示。对于一个给定的数据段 D，发送方要选择 r 个附加比特 R，并将它们附加到 D 上，使得得到的 $d+r$ 比特模式（被解释为一个二进制数）用模 2 算术恰好能被 G 整除（即没有余数）。用 CRC 进行差错检测的过程因此很简单：接收方用 G 去除接收到的 $d+r$ 比特。如果余数为非零，接收方知道出现了差错；否则认为数据正确而被接收。

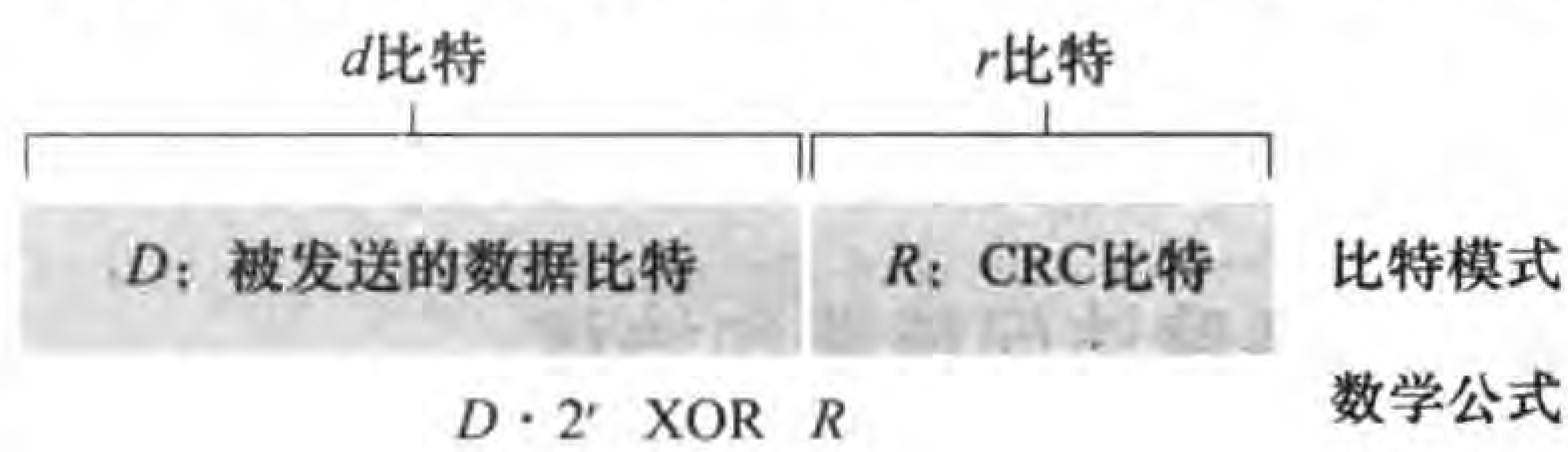


图 6-6 CRC

所有 CRC 计算采用模 2 算术来做，在加法中不进位，在减法中不借位。这意味着加法和减法是相同的，而且这两种操作等价于操作数的按位异或（XOR）。因此，举例来说：

$$1011 \text{ XOR } 0101 = 1110$$
$$1001 \text{ XOR } 1101 = 0100$$

类似地，我们还会有：

$$1011 - 0101 = 1110$$
$$1001 - 1101 = 0100$$

除了所需的加法或减法操作没有进位或借位外，乘法和除法与在二进制算术中是相同的。如在通常的二进制算术中那样，乘以 2^k 就是以一种比特模式左移 k 个位置。这样，给定 D 和 R，$D \cdot 2^r \text{XOR } R$ 产生如图 6-6 所示的 $d+r$ 比特模式。在下面的讨论中，我们将利用图 6-6 中这种 $d+r$ 比特模式的代数特性。

现在我们回到发送方怎样计算 R 这个关键问题上来。前面讲过，我们要求出 R 使得对于 n 有：

$$D \cdot 2^r \text{ XOR } R = nG$$

也就是说，我们要选择 R 使得 G 能够除以 $D \cdot 2^r \text{ XOR } R$ 而没有余数。如果我们对上述

等式的两边都用 R 异或（即用模 2 加，而没有进位），我们得到

$$D \cdot 2^r = nG \text{ XOR } R$$

这个等式告诉我们，如果我们用 G 来除 $D \cdot 2^r$，余数值刚好是 R。换句话说，我们可以这样计算 R：

$$R = \text{remainder}\ \frac{D \cdot 2^r}{G}$$

图 6-7 举例说明了在 $D=101110$，$d=6$，$G=1001$ 和 $r=3$ 的情况下的计算过程。在这种情况下传输的 9 个比特是 101110011。你应该自行检查一下这些计算，并核对一下 $D \cdot 2^r = 101011 \cdot G \text{ XOR } R$ 的确成立。

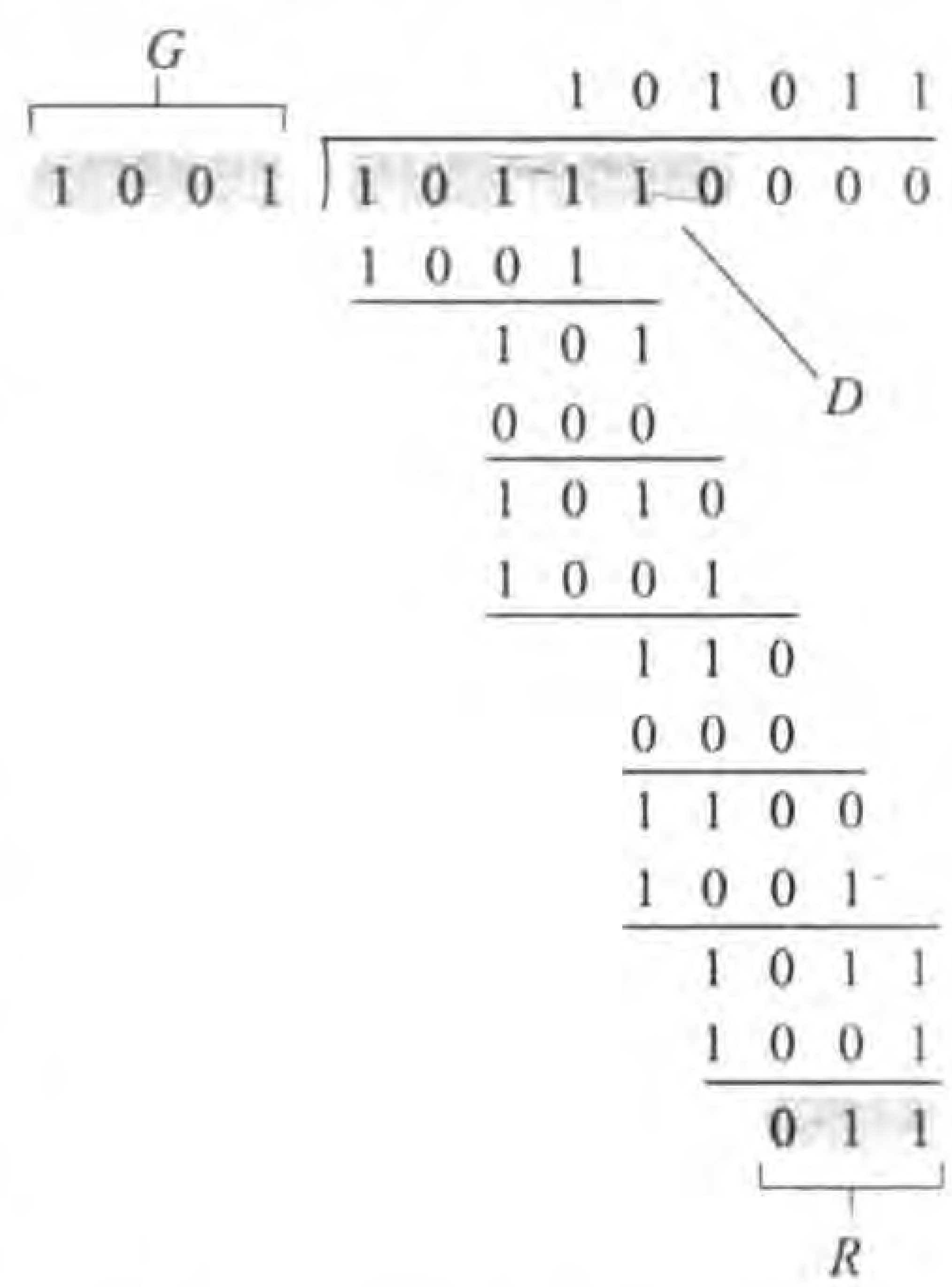


图 6-7　一个简单的 CRC 计算

国际标准已经定义了 8、12、16 和 32 比特生成多项式 G。CRC-32 32 比特的标准被多种链路级 IEEE 协议采用，使用的一个生成多项式是：

$$G_{CRC-32} = 100000100110000010001110110110111$$

每个 CRC 标准都能检测小于 $r+1$ 比特的突发差错。（这意味着所有连续的 r 比特或者更少的差错都可以检测到。）此外，在适当的假设下，长度大于 $r+1$ 比特的突发差错以概率 $1-0.5^r$ 被检测到。每个 CRC 标准也都能检测任何奇数个比特差错。有关 CRC 检测实现的讨论可参见 [Williams 1993]。CRC 编码甚至更强的编码所依据的理论超出了本书的范围。教科书 [Schwartz 1980] 对这个主题提供了很好的介绍。

6.3　多路访问链路和协议

在本章概述中，我们提到了有两种类型的网络链路：点对点链路和广播链路。**点对点链路**（point-to-point link）由链路一端的单个发送方和链路另一端的单个接收方组成。许多链路层协议都是为点对点链路设计的，如点对点协议（point-to-point protocol，PPP）和高级数据链路控制（high-level data link control，HDLC）就是两种这样的协议，我们将在本章后面涉及它们。第二种类型的链路是**广播链路**（broadcast link），它能够让多个发送和接收节点都连接到相同的、单一的、共享的广播信道上。这里使用术语“广播”是因为当任何一个节点传输一个帧时，信道广播该帧，每个其他节点都收到一个副本。以太网和无线局域网是广播链路层技术的例子。在本节，我们暂缓讨论特定的链路层协议，而先研究一个对链路层很重要的问题：如何协调多个发送和接收节点对一个共享广播信道的访问，这就是**多路访问问题**（multiple access problem）。广播信道通常用于局域网中，局域网是一个地理上集中在一座建筑物中（或者在一个公司，或者在大学校园）的网络。因此我们还将在本节后面考察一下多路访问信道是如何在局域网中使用的。

我们都很熟悉广播的概念，因为自电视发明以来就使用了这种通信方式。但是传统的电视是一种一个方向的广播（即一个固定的节点向许多接收节点传输），而计算机网络广播信道上的节点既能够发送也能够接收。也许对广播信道的一个更有人情味的类比是鸡尾酒会，在那里许多人聚集在一个大房间里（空气为提供广播的媒体）谈论和倾听。第二个

切题的类比是许多读者都很熟悉的地方，即一间教室，在那里老师们和同学们同样共享相同的、单一的广播媒体。在这两种场景下，一个中心问题是确定谁以及在什么时候获得说话权力（也就是向信道传输）。作为人类，为了共享这种广播信道，我们已经演化得到了一个精心设计的协议集了：

"给每个人一个讲话的机会。"

"该你讲话时你才说话。"

"不要一个人独占整个谈话。"

"如果有问题请举手。"

"当有人讲话时不要打断。"

"当其他人讲话时不要睡觉。"

计算机网络有类似的协议，也就是所谓的**多路访问协议**（multiple access protocol），即节点通过这些协议来规范它们在共享的广播信道上的传输行为。如图6-8所示，在各种各样的网络环境下需要多路访问协议，包括有线和无线接入网，以及卫星网络。尽管从技术上讲每个节点通过它的适配器访问广播信道，但在本节中我们将把节点作为发送和接收设备。在实践中，数以百计或者甚至数以千计个节点能够通过一个广播信道直接通信。

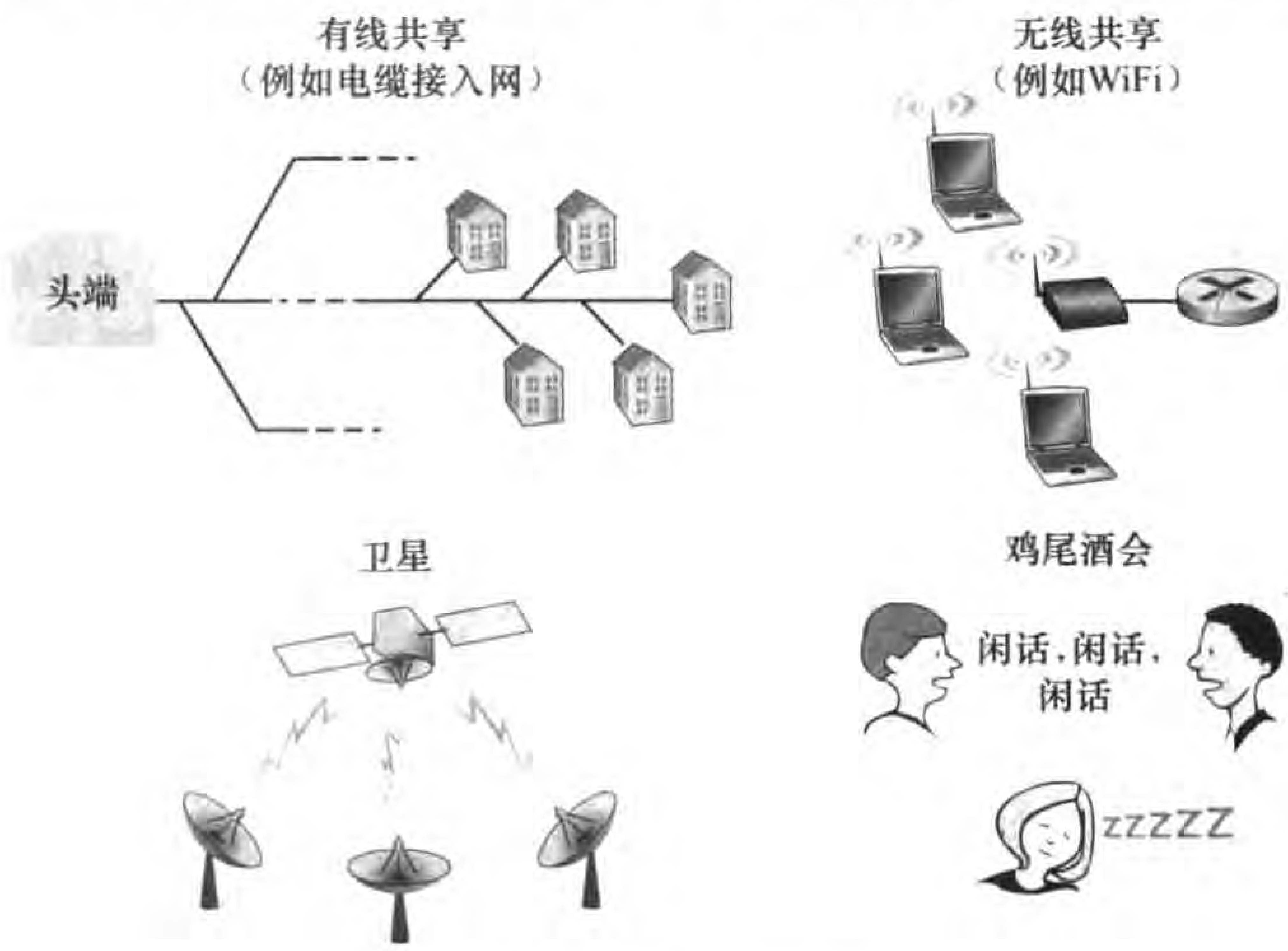


图6-8　多种多路访问信道

因为所有的节点都能够传输帧，所以多个节点可能会同时传输帧。当发生这种情况时，所有节点同时接到多个帧；这就是说，传输的帧在所有的接收方处**碰撞**（collide）了。通常，当碰撞发生时，没有一个接收节点能够有效地获得任何传输的帧；在某种意义下，碰撞帧的信号纠缠在一起。因此，涉及此次碰撞的所有帧都丢失了，在碰撞时间间隔中的广播信道被浪费了。显然，如果许多节点要频繁地传输帧，许多传输将导致碰撞，广播信道的大量带宽将被浪费掉。

当多个节点处于活跃状态时，为了确保广播信道执行有用的工作，以某种方式协调活跃节点的传输是必要的。这种协调工作由多路访问协议负责。在过去的40年中，已经有上千篇文章和上百篇博士论文研究过多路访问协议；有关这部分工作前20年来的一个内容丰富的综述见［Rom 1990］。此外，由于新类型链路尤其是新的无线链路不断出现，在多路访问协议方面研究的活跃状况仍在继续。

这些年来，在大量的链路层技术中已经实现了几十种多路访问协议。尽管如此，我们能够将任何多路访问协议划分为3种类型之一：**信道划分协议**（channel partitioning protocol），**随机接入协议**（random access protocol）和**轮流协议**（taking-turns protocol）。我们将在后续的3个小节中讨论这几类多路访问协议。

在结束概述之前，我们给出下列条件。在理想情况下，对于速率为 R bps 的广播信道，多路访问协议应该具有以下所希望的特性：

1）当仅有一个节点发送数据时，该节点具有 R bps 的吞吐量；

2）当有 M 个节点发送数据时，每个节点吞吐量为 R/M bps。这不必要求 M 个节点中的每一个节点总是有 R/M 的瞬间速率，而是每个节点在一些适当定义的时间间隔内应该有 R/M 的平均传输速率。

3）协议是分散的；这就是说不会因某主节点故障而使整个系统崩溃。

4）协议是简单的，使实现不昂贵。

6.3.1　信道划分协议

我们前面在1.3节讨论过，时分多路复用（TDM）和频分多路复用（FDM）是两种能够用于在所有共享信道节点之间划分广播信道带宽的技术。举例来说，假设一个支持 N 个节点的信道且信道的传输速率为 R bps。TDM 将时间划分为**时间帧**（time frame），并进一步划分每个时间帧为 N 个**时隙**（slot）。（不应当把 TDM 时间帧与在发送和接收适配器之间交换的链路层数据单元相混淆，后者也被称为帧。为了减少混乱，在本小节中我们将链路层交换的数据单元称为分组。）然后把每个时隙分配给 N 个节点中的一个。无论何时某个节点在有分组要发送的时候，它在循环的 TDM 帧中指派给它的时隙内传输分组比特。通常，选择的时隙长度应使一个时隙内能够传输单个分组。图6-9表示一个简单的4个节点的 TDM 例子。再回到我们的鸡尾酒会类比中，一个采用 TDM 规则的鸡尾酒会将允许每个聚会客人在固定的时间段发言，然后再允许另一个聚会客人发言同样时长，以此类推。一旦每个人都有了说话机会，将不断重复着这种模式。

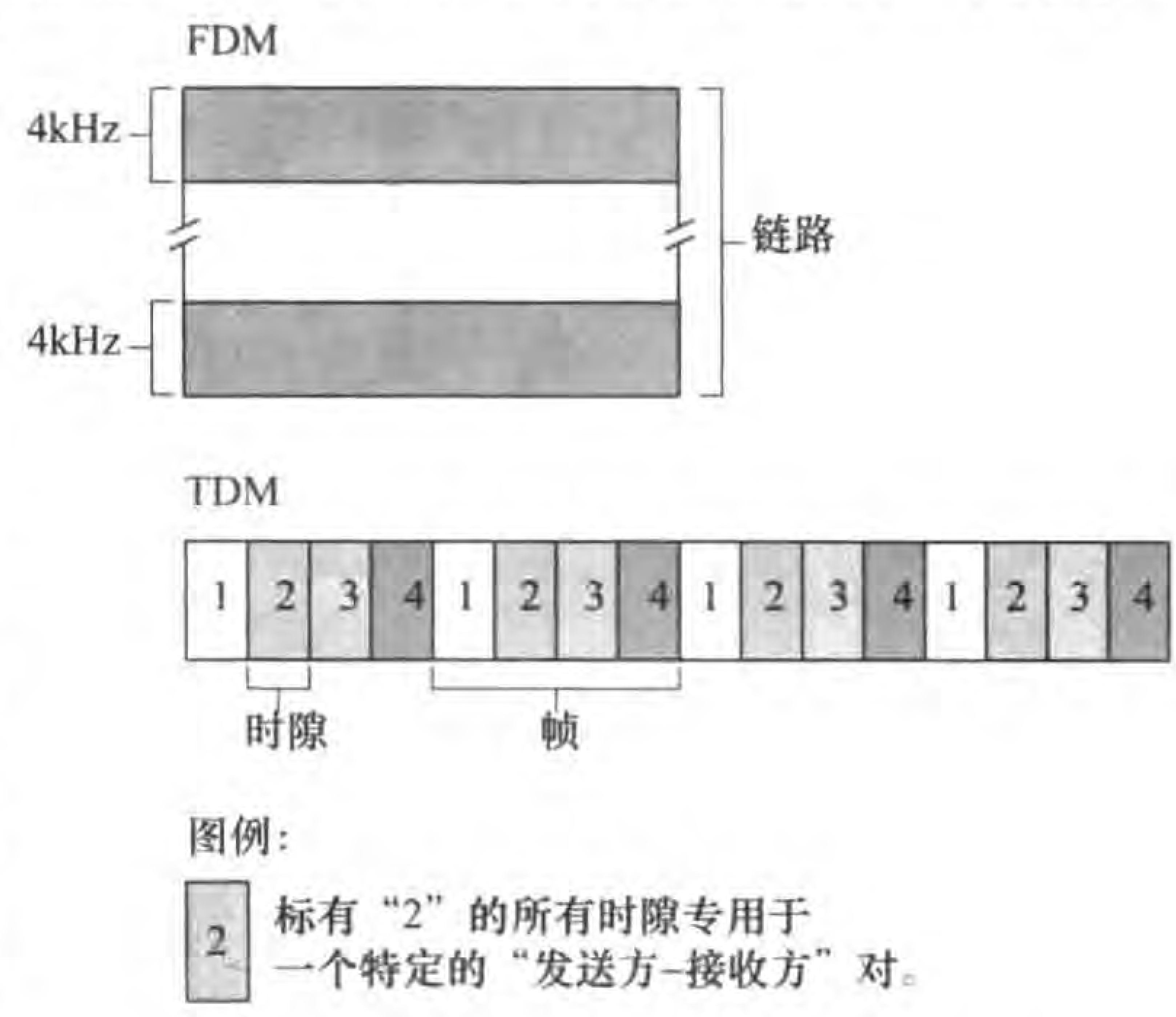


图6-9　一个4节点的 TDM 与 FDM 的例子

TDM 是有吸引力的，因为它消除了碰撞而且非常公平：每个节点在每个帧时间内得到了专用的传输速率 R/N bps。然而它有两个主要缺陷。首先，节点被限制于 R/N bps 的平均速率，即使当它是唯一有分组要发送的节点时。其次，节点必须总是等待它在传输序列中的轮次，即我们再次看到，即使它是唯一一个有帧要发送的节点。想象一下某聚会客人是唯一一个有话要说的人的情形（并且想象一下这种十分罕见的情况，即酒会上所有的人都想听某一个人说话）。显然，一种多路访问协议用于这个特殊聚会时，TDM 是一种很糟的选择。

TDM 在时间上共享广播信道，而 FDM 将 R bps 信道划分为不同的频段（每个频段

具有 R/N 带宽)，并把每个频率分配给 N 个节点中的一个。因此 FDM 在单个较大的 R bps 信道中创建了 N 个较小的 R/N bps 信道。FDM 也有 TDM 同样的优点和缺点。它避免了碰撞，在 N 个节点之间公平地划分了带宽。然而，FDM 也有 TDM 所具有的主要缺点，也就是限制一个节点只能使用 R/N 的带宽，即使当它是唯一一个有分组要发送的节点时。

第三种信道划分协议是**码分多址**（Code Division Multiple Access，CDMA)。TDM 和 FDM 分别为节点分配时隙和频率，而 CDMA 对每个节点分配一种不同的编码。然后每个节点用它唯一的编码来对它发送的数据进行编码。如果精心选择这些编码，CDMA 网络具有一种奇妙的特性，即不同的节点能够同时传输，并且它们各自相应的接收方仍能正确接收发送方编码的数据比特（假设接收方知道发送方的编码)，而不在乎其他节点的干扰传输。CDMA 已经在军用系统中使用了一段时间（由于它的抗干扰特性)，目前已经广泛地用于民用，尤其是蜂窝电话中。因为 CDMA 的使用与无线信道紧密相关，所以我们将把有关 CDMA 技术细节的讨论留到第 7 章。此时，我们知道 CDMA 编码类似于 TDM 中的时隙和 FDM 中的频率，能分配给多路访问信道的用户就可以了。

6.3.2 随机接入协议

第二大类多访问协议是随机接入协议。在随机接入协议中，一个传输节点总是以信道的全部速率（即 R bps）进行发送。当有碰撞时，涉及碰撞的每个节点反复地重发它的帧（也就是分组)，到该帧无碰撞地通过为止。但是当一个节点经历一次碰撞时，它不必立刻重发该帧。相反，*它在重发该帧之前等待一个随机时延*。涉及碰撞的每个节点独立地选择随机时延。因为该随机时延是独立地选择的，所以下述现象是有可能的：这些节点之一所选择的时延充分小于其他碰撞节点的时延，并因此能够无碰撞地将它的帧在信道中发出。

文献中描述的随机接入协议即使没有上百种也有几十种［Rom 1990；Bertsekas 1991］。在本节中，我们将描述一些最常用的随机接入协议，即 ALOHA 协议［Abramson 1970；Abramson 1985；Abramson 2009］和载波侦听多路访问（CSMA）协议［Kleinrock 1975b］。以太网［Metcalfe 1976］是一种流行并广泛部署的 CSMA 协议。

1. 时隙 ALOHA

我们以最简单的随机接入协议之一——时隙 ALOHA 协议，开始我们对随机接入协议的学习。在对时隙 ALOHA 的描述中，我们做下列假设：

- 所有帧由 L 比特组成。
- 时间被划分成长度为 L/R 秒的时隙（这就是说，一个时隙等于传输一帧的时间)。
- 节点只在时隙起点开始传输帧。
- 节点是同步的，每个节点都知道时隙何时开始。
- 如果在一个时隙中有两个或者更多个帧碰撞，则所有节点在该时隙结束之前检测到该碰撞事件。

令 p 是一个概率，即一个在 0 和 1 之间的数。在每个节点中，时隙 ALOHA 的操作是简单的：

- 当节点有一个新帧要发送时，它等到下一个时隙开始并在该时隙传输整个帧。
- 如果没有碰撞，该节点成功地传输它的帧，从而不需要考虑重传该帧。（如果该节

点有新帧，它能够为传输准备一个新帧。）

- 如果有碰撞，该节点在时隙结束之前检测到这次碰撞。该节点以概率 p 在后续的每个时隙中重传它的帧，直到该帧被无碰撞地传输出去。

我们说以概率 p 重传，是指某节点有效地投掷一个有偏倚的硬币；硬币正面事件对应着重传，而重传出现的概率为 p。硬币反面事件对应着“跳过这个时隙，在下个时隙再掷硬币”；这个事件以概率 $(1-p)$ 出现。所有涉及碰撞的节点独立地投掷它们的硬币。

时隙 ALOHA 看起来有很多优点。与信道划分不同，当某节点是唯一活跃的节点时（一个节点如果有帧要发送就认为它是活跃的），时隙 ALOHA 允许该节点以全速 R 连续传输。时隙 ALOHA 也是高度分散的，因为每个节点检测碰撞并独立地决定什么时候重传。（然而，时隙 ALOHA 的确需要在节点中对时隙同步；我们很快将讨论 ALOHA 协议的一个不分时隙的版本以及 CSMA 协议，这两种协议都不需要这种同步。）时隙 ALOHA 也是一个极为简单的协议。

当只有一个活跃节点时，时隙 ALOHA 工作出色，但是当有多个活跃节点时效率又将如何呢？这里有两个可能要考虑的效率问题。首先，如在图 6-10 中所示，当有多个活跃节点时，一部分时隙将有碰撞，因此将被“浪费”掉了。第二个考虑是，时隙的另一部分将是空闲的，因为所有活跃节点由于概率传输策略会节制传输。唯一“未浪费的”时隙是那些刚好有一个节点传输的时隙。刚好有一个节点传输的时隙称为一个**成功时隙**（successful slot）。时隙多路访问协议的**效率**（efficiency）定义为：当有大量的活跃节点且每个节点总有大量的帧要发送时，长期运行中成功时隙的份额。注意到如果不使用某种形式的访问控制，而且每个节点都在每次碰撞之后立即重传，这个效率将为零。时隙 ALOHA 显然增加了它的效率，使之大于零，但是效率增加了多少呢？

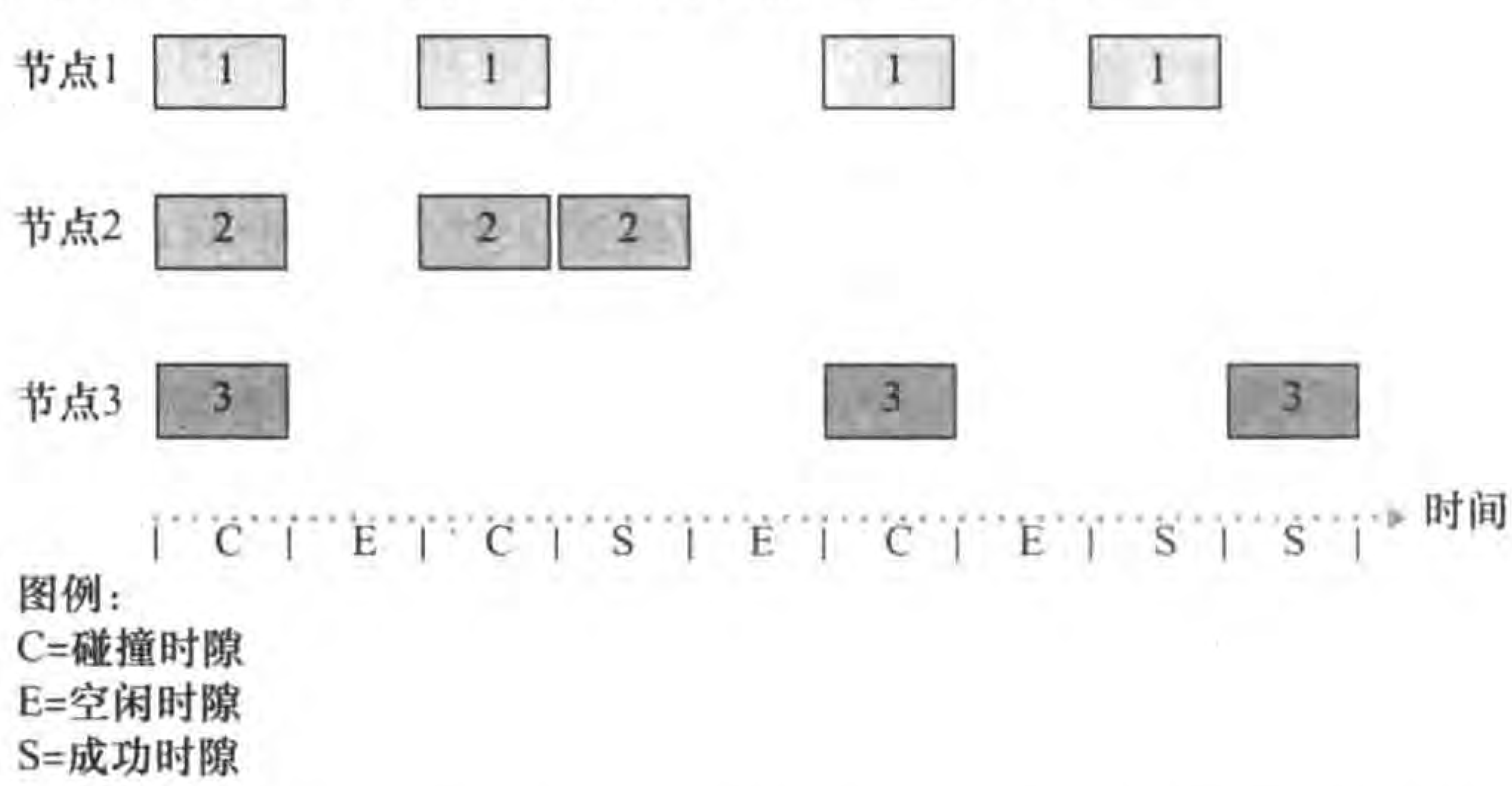


图 6-10 节点 1、2 和 3 在第一个时隙碰撞。节点 2 最终在第 4 个时隙成功，节点 1 在第 8 个时隙成功，节点 3 在第 9 个时隙成功

现在我们继续概要讨论时隙 ALOHA 最大效率的推导过程。为了保持该推导简单，我们对协议做了一点修改，假设每个节点试图在每个时隙以概率 p 传输一帧。（这就是说，我们假设每个节点总有帧要发送，而且节点对新帧和已经经历一次碰撞的帧都以概率 p 传输。）假设有 N 个节点。则一个给定时隙是成功时隙的概率为节点之一传输而余下的 $N-1$ 个节点不传输的概率。一个给定节点传输的概率是 p；剩余节点不传输的概率是 $(1-p)^{N-1}$。因此，一个给定节点成功传送的概率是 $p(1-p)^{N-1}$。因为有 N 个节点，任意一个节点成功传送的概率是 $Np(1-p)^{N-1}$。

因此，当有 N 个活跃节点时，时隙 ALOHA 的效率是 $Np(1-p)^{N-1}$。为了获得 N 个活跃节点的最大效率，我们必须求出使这个表达式最大化的 p^*。（对这个推导的一个大体描述参见课后习题。）而且对于大量活跃节点，为了获得最大效率，当 N 趋于无穷时，我们取 $Np^*(1-p^*)^{N-1}$ 的极限。（同样参见课后习题。）在完成这些计算之后，我们会发现这个协议的最大效率为 $1/e=0.37$。这就是说，当有大量节点有很多帧要传输时，则（最多）仅有 37% 的时隙做有用的工作。因此该信道有效传输速率不是 R bps，而仅为 $0.37R$ bps！相似的分析还表明 37% 的时隙是空闲的，26% 的时隙有碰撞。试想一个蹩脚的网络管理员购买了一个 100Mbps 的时隙 ALOHA 系统，希望能够使用网络在大量的用户之间以总计速率如 80Mbps 来传输数据。尽管这个信道能够以信道的全速 100Mbps 传输一个给定的帧，但从长时间范围看，该信道的成功吞吐量将小于 37Mbps。

2. ALOHA

时隙 ALOHA 协议要求所有的节点同步它们的传输，以在每个时隙开始时开始传输。第一个 ALOHA 协议［Abramson 1970］实际上是一个非时隙、完全分散的协议。在纯 ALOHA 中，当一帧首次到达（即一个网络层数据报在发送节点从网络层传递下来），节点立刻将该帧完整地传输进广播信道。如果一个传输的帧与一个或多个传输经历了碰撞，这个节点将立即（在完全传输完它的碰撞帧之后）以概率 p 重传该帧。否则，该节点等待一个帧传输时间。在此等待之后，它则以概率 p 传输该帧，或者以概率 $1-p$ 在另一个帧时间等待（保持空闲）。

为了确定纯 ALOHA 的最大效率，我们关注某个单独的节点。我们的假设与在时隙 ALOHA 分析中所做的相同，取帧传输时间为时间单元。在任何给定时间，某节点传输一个帧的概率是 p。假设该帧在时刻 t_0 开始传输。如图 6-11 中所示，为了使该帧能成功地传输，在时间间隔 $[t_0-1, t_0]$ 中不能有其他节点开始传输。这种传输将与节点 i 的帧传输起始部分相重叠。所有其他节点在这个时间间隔不开始传输的概率是 $(1-p)^{N-1}$。类似地，当节点 i 在传输时，其他节点不能开始传输，因为这种传输将与节点 i 传输的后面部分相重叠。所有其他节点在这个时间间隔不开始传输的概率也是 $(1-p)^{N-1}$。因此，一个给定的节点成功传输一次的概率是 $p(1-p)^{2(N-1)}$。通过与时隙 ALOHA 情况一样来取极限，我们求得纯 ALOHA 协议的最大效率仅为 $1/(2e)$，这刚好是时隙 ALOHA 的一半。这就是完全分散的 ALOHA 协议所要付出的代价。

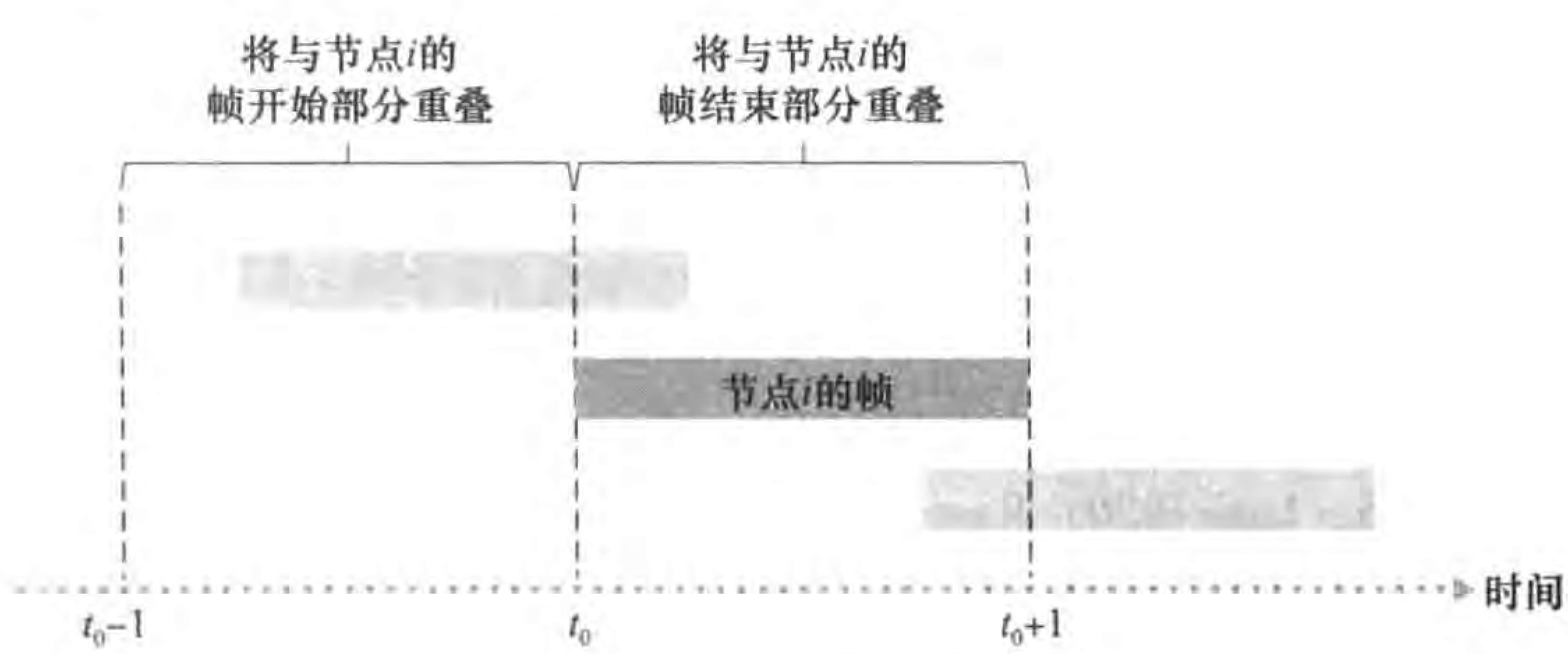


图 6-11　纯 ALOHA 中的干扰传输

历史事件

Norm Abramson 和 ALOHAnet

Norm Abramson 是一名有博士学位的工程师，对冲浪运动很有激情，而且对分组交换很感兴趣。这些兴趣的结合使他在 1969 年到了夏威夷大学。夏威夷是由许多巨大的岛屿组成的，安装和运营基于陆地的网络是困难的。当不冲浪的时候，Abramson 思考如何设计一种在无线信道上完成分组交互的网络。他设计的网络有一个中心主机和几个分散在夏威夷各个岛上的二级节点。该网络有两个信道，每个信道使用不同的频段。下行链路信道从中心主机向二级主机广播分组；上行信道从二级主机向中心主机发送分组。除了发送信息分组，中心主机还在下行信道上对从二级主机成功接收到的每个分组发送确认。

因为二级主机以分散的方式传输分组，在上行信道上出现碰撞是不可避免的。这个观察导致 Abramson 设计了如本章所描述的那种纯 ALOHA 协议。在 1970 年，通过不断从 ARPA 获得的资助，Abramson 将他的 ALOHAnet 与 ARPAnet 相连。Abramson 的工作是很重要的，不仅因为它是无线分组网络的第一个例子，而且因为它激励了 Bob Metcalfe。几年之后，Metcalfe 修改了 ALOHA 协议，创造了 CSMA/CD 协议和以太网局域网。

3. 载波侦听多路访问（CSMA）

在时隙和纯 ALOHA 中，一个节点传输的决定独立于连接到这个广播信道上的其他节点的活动。特别是，一个节点不关心在它开始传输时是否有其他节点碰巧在传输，而且即使有另一个节点开始干扰它的传输也不会停止传输。在我们的鸡尾酒会类比中，ALOHA 协议非常像一个粗野的聚会客人，他喋喋不休地讲话而不顾是否其他人在说话。作为人类，我们有人类的协议，它要求我们不仅要更为礼貌，而且在谈话中要减少与他人“碰撞”的时间，从而增加我们谈话中交流的数据量。具体而言，有礼貌的人类谈话有两个重要的规则：

- *说话之前先听*。如果其他人正在说话，等到他们说完话为止。在网络领域中，这被称为**载波侦听**（carrier sensing），即一个节点在传输前先听信道。如果来自另一个节点的帧正向信道上发送，节点则等待直到检测到一小段时间没有传输，然后开始传输。
- *如果与他人同时开始说话，停止说话*。在网络领域中，这被称为**碰撞检测**（collision detection），即当一个传输节点在传输时一直在侦听此信道。如果它检测到另一个节点正在传输干扰帧，它就停止传输，在重复“侦听－当空闲时传输”循环之前等待一段随机时间。

这两个规则包含在**载波侦听多路访问**（Carrier Sense Multiple Access，CSMA）和**具有碰撞检测的 CSMA**（CSMA with Collision Detection，CSMA/CD）协议族中［Kleinrock 1975b；Metcalfe 1976；Lam 1980；Rom 1990］。人们已经提出了 CSMA 和 CSMA/CD 的许多变种。这里，我们将考虑一些 CSMA 和 CSMA/CD 最重要的和基本的特性。

关于 CSMA 你可能要问的第一个问题是，如果所有的节点都进行载波侦听了，为什么当初会发生碰撞？毕竟，某节点无论何时侦听到另一个节点在传输，它都会停止传输。对于这个问题的答案最好能够用时空图来说明［Molle 1987］。图 6-12 显示了连接到一个线

状广播总线的4个节点（A、B、C、D）的时空图。横轴表示每个节点在空间的位置；纵轴表示时间。

在时刻 t_0，节点B侦听到信道是空闲的，因为当前没有其他节点在传输。因此节点B开始传输，沿着广播媒体在两个方向上传播它的比特。图6-12中B的比特随着时间的增加向下传播，这表明B的比特沿着广播媒体传播所实际需要的时间不是零（虽然以接近光的速度）。在时刻 t_1（$t_1 > t_0$），节点D有一个帧要发送。尽管节点B在时刻 t_1 正在传输，但B传输的比特还没有到达D，因此D在 t_1 侦听到信道空闲。根据CSMA协议，从而D开始传输它的帧。一个短暂的时间之后，B的传输开始在D干扰D的传输。从图6-12中可以看出，显然广播信道的端到端**信道传播时延**（channel propagation delay）（信号从一个节点传播到另一个节点所花费的时间）在决定其性能方面起着关键的作用。该传播时延越长，载波侦听节点不能侦听到网络中另一个节点已经开始传输的机会就越大。

4. 具有碰撞检测的载波侦听多路访问（CSMA/CD）

在图6-12中，节点没有进行碰撞检测；即使已经出现了碰撞，B和D都将继续完整地传输它们的帧。当某节点执行碰撞检测时，一旦它检测到碰撞将立即停止传输。图6-13表示了和图6-12相同的情况，只是这两个节点在检测到碰撞后很短的时间内都放弃了它们的传输。显然，在多路访问协议中加入碰撞检测，通过不传输一个无用的、（由来自另一个节点的帧干扰）损坏的帧，将有助于改善协议的性能。

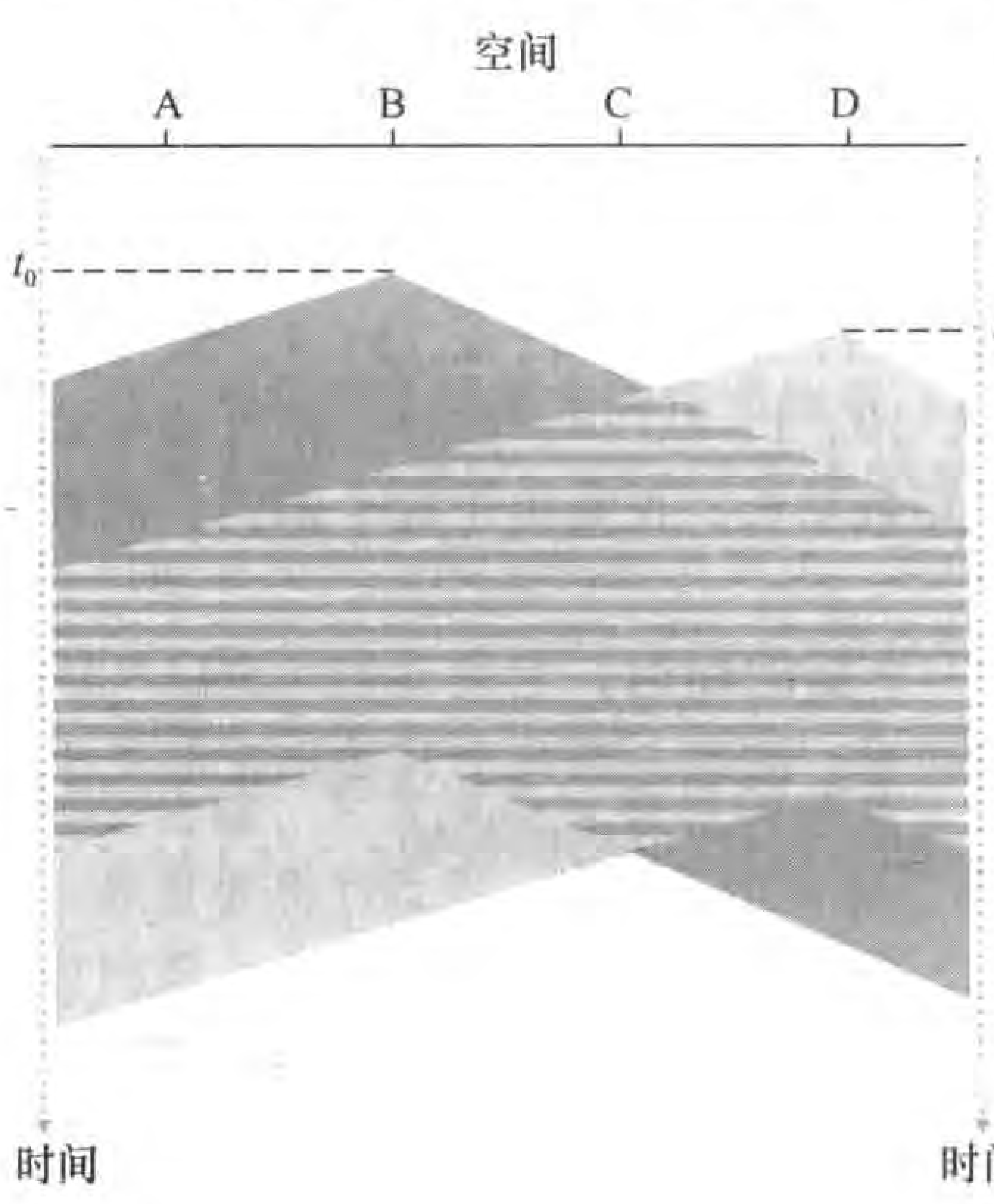


图6-12 发生碰撞传输的两个CSMA节点的时空图

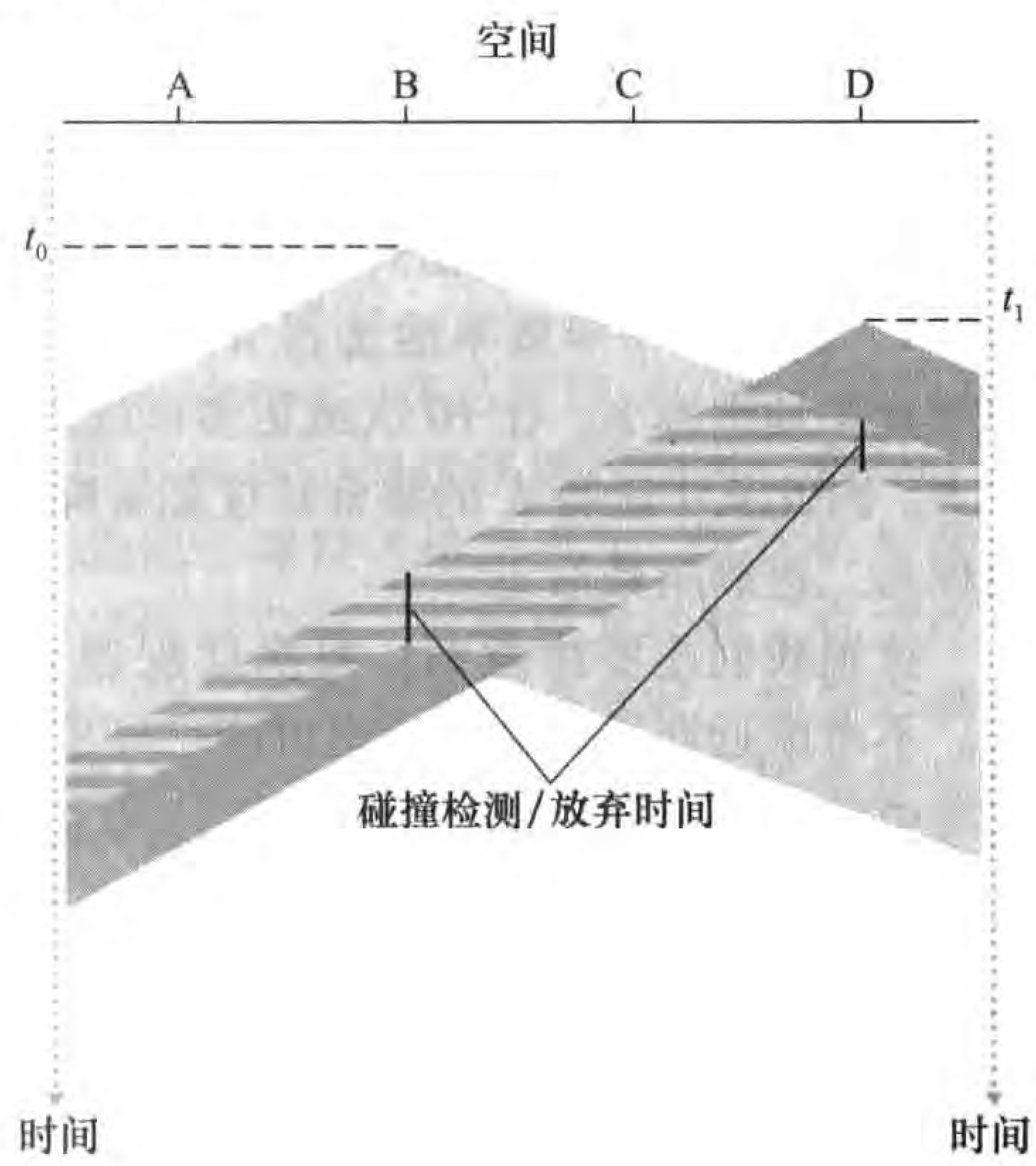


图6-13 具有碰撞检测的CSMA

在分析CSMA/CD协议之前，我们现在从与广播信道相连的适配器（在节点中）的角度总结它的运行：

1）适配器从网络层一条获得数据报，准备链路层帧，并将其放入帧适配器缓存中。

2）如果适配器侦听到信道空闲（即无信号能量从信道进入适配器），它开始传输帧。在另一方面，如果适配器侦听到信道正在忙，它将等待，直到侦听到没有信号能量时才开始传输帧。

3）在传输过程中，适配器监视来自其他使用该广播信道的适配器的信号能量的存在。

4）如果适配器传输整个帧而未检测到来自其他适配器的信号能量，该适配器就完成了该帧。在另一方面，如果适配器在传输时检测到来自其他适配器的信号能量，它中止传输（即它停止了传输帧）。

5）中止传输后，适配器等待一个随机时间量，然后返回步骤2。

等待一个随机（而不是固定）的时间量的需求是明确的——如果两个节点同时传输帧，然后这两个节点等待相同固定的时间量，它们将持续碰撞下去。但选择随机回退时间的时间间隔多大为好呢？如果时间间隔大而碰撞节点数量小，在重复“侦听－当空闲时传输”的步骤前，节点很可能等待较长的时间（使信道保持空闲）。在另一方面，如果时间间隔小而碰撞节点数量大，很可能选择的随机值将几乎相同，传输节点将再次碰撞。我们希望时间间隔应该这样：当碰撞节点数量较少时，时间间隔较短；当碰撞节点数量较大时，时间间隔较长。

用于以太网以及DOCSIS电缆网络多路访问协议［DOCSIS 2011］中的**二进制指数后退**（binary exponential backoff）算法，简练地解决了这个问题。特别是，当传输一个给定帧时，在该帧经历了一连串的 n 次碰撞后，节点随机地从 $\{0, 1, 2, \cdots, 2^n-1\}$ 中选择一个 K 值。因此，一个帧经历的碰撞越多，K 选择的间隔越大。对于以太网，一个节点等待的实际时间量是 $K \cdot 512$ 比特时间（即发送512比特进入以太网所需时间量的 K 倍），n 能够取的最大值在10以内。

我们看一个例子。假设一个适配器首次尝试传输一个帧，并在传输中它检测到碰撞。然后该节点以概率0.5选择 $K=0$，以概率0.5选择 $K=1$。如果该节点选择 $K=0$，则它立即开始侦听信道。如果这个适配器选择 $K=1$，它在开始“侦听－当空闲时传输”。周期前等待512比特时间（例如对于100Mbps以太网来说为5.12ms）。在第2次碰撞之后，从 $\{0, 1, 2, 3\}$ 中等概率地选择 K。在第3次碰撞之后，从 $\{0, 1, 2, 3, 4, 5, 6, 7\}$ 中等概率地选择 K。在10次或更多次碰撞之后，从 $\{0, 1, 2, \cdots, 1023\}$ 中等概率地选择 K。因此从中选择 K 的集合长度随着碰撞次数呈指数增长；正是由于这个原因，该算法被称为二进制指数后退。

这里我们还要注意到，每次适配器准备传输一个新的帧时，它要运行CSMA/CD算法。不考虑近期过去的时间内可能已经发生的任何碰撞。因此，当几个其他适配器处于指数后退状态时，有可能一个具有新帧的节点能够立刻插入一次成功的传输。

5. CSMA/CD效率

当只有一个节点有一个帧发送时，该节点能够以信道全速率进行传输（例如10Mbps、100Mbps或者1Gbps）。然而，如果很多节点都有帧要发送，信道的有效传输速率可能会小得多。我们将**CSMA/CD效率**（efficiency of CSMA/CD）定义为：当有大量的活跃节点，且每个节点有大量的帧要发送时，帧在信道中无碰撞地传输的那部分时间在长期运行时间中所占的份额。为了给出效率的一个闭式的近似表示，令 d_{prop} 表示信号能量在任意两个适配器之间传播所需的最大时间。令 d_{trans} 表示传输一个最大长度的以太网帧的时间（对于10Mbps的以太网，该时间近似为1.2毫秒）。CSMA/CD效率的推导超出了本书的范围（见［Lam 1980］和［Bertsekas 1991］）。这里我们只是列出下面的近似式：

$$\text{效率} = \frac{1}{1 + 5d_{prop}/d_{trans}}$$

从这个公式我们看到，当 d_{prop} 接近0时，效率接近1。这和我们的直觉相符，如果传

播时延是 0，碰撞的节点将立即中止而不会浪费信道。同时，当 d_{trans} 变得很大时，效率也接近于 1。这也和直觉相符，因为当一个帧取得了信道时，它将占有信道很长时间；因此信道在大多数时间都会有效地工作。

6.3.3 轮流协议

前面讲过多路访问协议的两个理想特性是：①当只有一个节点活跃时，该活跃节点具有 *R* bps 的吞吐量；②当有 *M* 个节点活跃时，每个活跃节点的吞吐量接近 *R/M* bps。ALOHA 和 CSMA 协议具备第一个特性，但不具备第二个特性。这激发研究人员创造另一类协议，也就是**轮流协议**（taking-turns protocol）。和随机接入协议一样，有几十种轮流协议，其中每一个协议又都有很多变种。这里我们要讨论两种比较重要的协议。第一种是**轮询协议**（polling protocol）。轮询协议要求这些节点之一要被指定为主节点。主节点以循环的方式**轮询**（poll）每个节点。特别是，主节点首先向节点 1 发送一个报文，告诉它（节点 1）能够传输的帧的最多数量。在节点 1 传输了某些帧后，主节点告诉节点 2 它（节点 2）能够传输的帧的最多数量。（主节点能够通过观察在信道上是否缺乏信号，来决定一个节点何时完成了帧的发送。）上述过程以这种方式继续进行，主节点以循环的方式轮询了每个节点。

轮询协议消除了困扰随机接入协议的碰撞和空时隙，这使得轮询取得高得多的效率。但是它也有一些缺点。第一个缺点是该协议引入了轮询时延，即通知一个节点"它可以传输"所需的时间。例如，如果只有一个节点是活跃的，那么这个节点将以小于 *R* bps 的速率传输，因为每次活跃节点发送了它最多数量的帧时，主节点必须依次轮询每一个非活跃的节点。第二个缺点可能更为严重，就是如果主节点有故障，整个信道都变得不可操作。我们在本节学习的 802.15 协议和蓝牙协议就是轮询协议的例子。

第二种轮流协议是**令牌传递协议**（token-passing protocol）。在这种协议中没有主节点。一个称为**令牌**（token）的小的特殊帧在节点之间以某种固定的次序进行交换。例如，节点 1 可能总是把令牌发送给节点 2，节点 2 可能总是把令牌发送给节点 3，而节点 *N* 可能总是把令牌发送给节点 1。当一个节点收到令牌时，仅当它有一些帧要发送时，它才持有这个令牌；否则，它立即向下一个节点转发该令牌。当一个节点收到令牌时，如果它确实有帧要传输，它发送最大数目的帧数，然后把令牌转发给下一个节点。令牌传递是分散的，并有很高的效率。但是它也有自己的一些问题。例如，一个节点的故障可能会使整个信道崩溃。或者如果一个节点偶然忘记了释放令牌，则必须调用某些恢复步骤使令牌返回到循环中来。经过多年，人们已经开发了许多令牌传递协议，包括光纤分布式数据接口（FDDI）协议[Jain 1994] 和 IEEE 802.5 令牌环协议 [IEEE 802.5 2012]，每一种都必须解决这些和其他一些棘手的问题。

6.3.4 DOCSIS：用于电缆因特网接入的链路层协议

在前面 3 小节中，我们已经学习了 3 大类多路访问协议：信道划分协议、随机接入协议和轮流协议。这里的电缆接入网将作为一种很好的学习案例，因为在电缆接入网中我们将看到这三类多路访问协议中的每一种！

1.2.1 节讲过，一个电缆接入网通常在电缆网头端将几千个住宅电缆调制解调器与一个**电缆调制解调器端接系统**（Cable Modem Termination System，CMTS）连接。**数据经电缆服务接口**（Data-Over-Cable Service Interface，CMTS）规范（DOCSIS）[DOCSIS 2011] 定

义了电缆数据网络体系结构及其协议。DOCSIS 使用 FDM 将下行（CMTS 到调制解调器）和上行（调制解调器到 CMTS）网络段划分为多个频率信道。每个下行信道宽 6MHz，每个信道具有大约 40Mbps 吞吐量（尽管这种数据率在实践中很少在电缆调制解调器中见到）；每个上行信道具有 6.4MHz 的最大信道带宽，并且最大的上行吞吐量约为 30Mbps。每个上行和下行信道均为广播信道。CMTS 在下行信道中传输的帧被所有在信道上做接收的电缆调制解调器接收到；然而因为仅有单一的 CMTS 在下行信道上传输，不存在多路访问问题。但在上行方向，存在着多个有趣的技术挑战，因为多个电缆调制解调器共享到 CMTS 的相同上行信道（频率），因此能够潜在地出现碰撞。

如图 6-14 所示，每条上行信道被划分为时间间隔（类似于 TDM），每个时间间隔包含一个微时隙序列，电缆调制解调器可在该微时隙中向 CMTS 传输。CMTS 显式地准许各个电缆调制解调器在特定的微时隙中进行传输。CMTS 在下行信道上通过发送称为 MAP 报文的控制报文，指定哪个电缆调制解调器（带有要发送的数据）能够在微时隙中传输由控制报文指定的时间间隔。由于微时隙明确分配给电缆调制解调器，故 CMTS 能够确保在微时隙中没有碰撞传输。

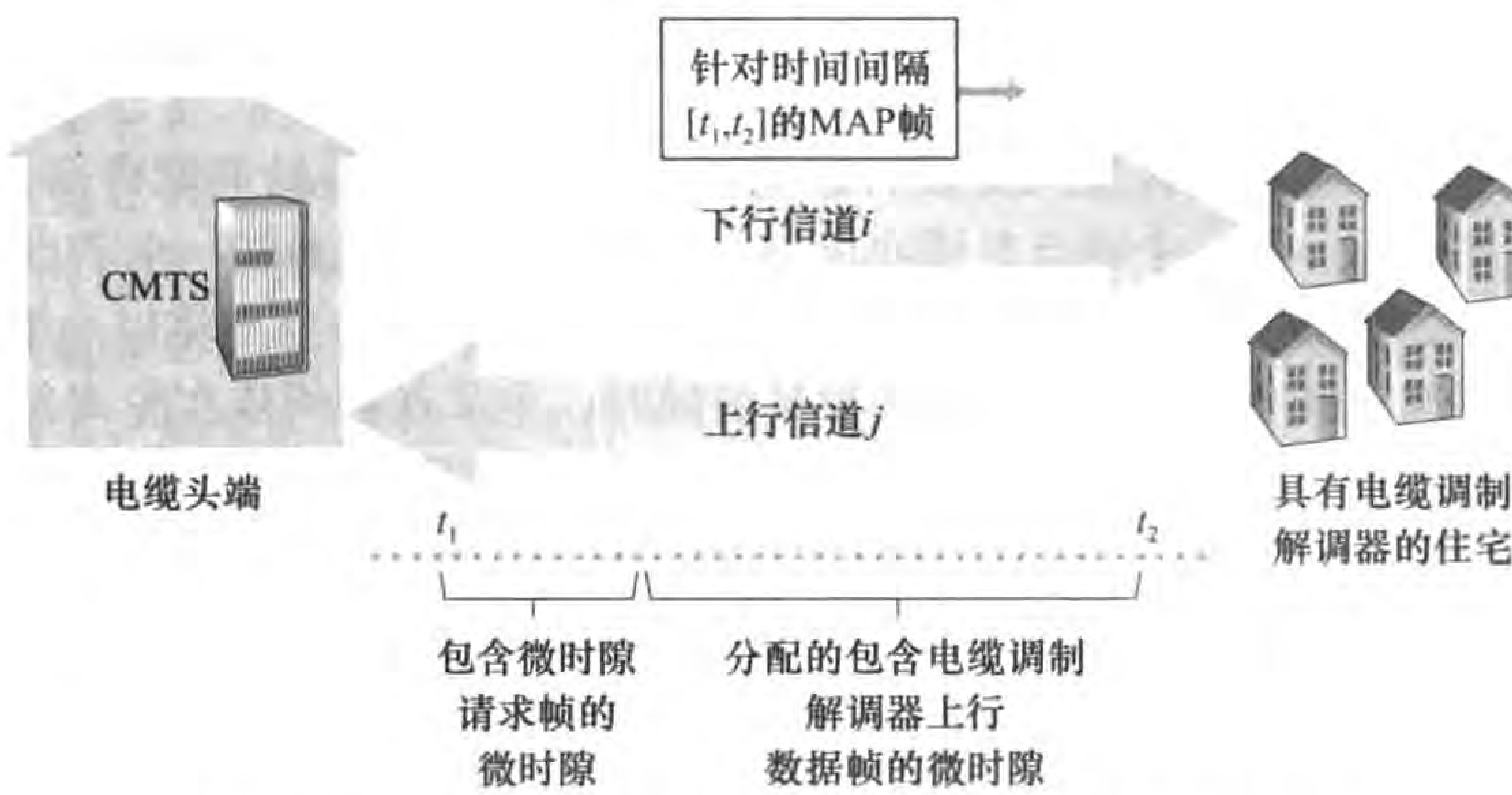


图 6-14　CMTS 和电缆调制解调器之间的上行和下行信道

但是 CMTS 一开始是如何知道哪个电缆调制解调器有数据要发送呢？通过让电缆调制解调器在专用于此目的的一组特殊的微时隙间隔内向 CMTS 发送微时隙请求帧来完成该任务，如图 6-14 所示。这些微时隙请求帧以随机接入方式传输，故可能相互碰撞。电缆调制解调器既不能侦听上行信道是否忙，也不能检测碰撞。相反，该电缆调制解调器如果没有在下一个下行控制报文中收到对请求分配的响应的话，就推断出它的微时隙请求帧经历了一次碰撞。当推断出一次碰撞，电缆调制解调器使用二进制指数回退将其微时隙请求帧延缓到以后的时隙重新发送。当在上行信道上有很少的流量，电缆调制解调器可能在名义上分配给微时隙请求帧的时隙内实际传输数据帧（因此避免不得不等待微时隙分配）。

因此，电缆接入网可作为应用多路访问协议（即 FDM、TDM、随机接入和集中分配时隙都用于一个网络中）的一个极好例子。

6.4　交换局域网

前面一节涉及了广播网络和多路访问协议，我们现在将注意力转向交换局域网。

图6-15显示了一个交换局域网连接了3个部门，两台服务器和一台与4台交换机连接的路由器。因为这些交换机运行在链路层，所以它们交换链路层帧（而不是网络层数据报），不识别网络层地址，不使用如RIP或OSPF这样的路由选择算法来确定通过第二层交换机网络的路径。我们马上就会看到，它们使用链路层地址而不是IP地址来转发链路层帧通过交换机网络。我们首先以讨论链路层寻址（6.4.1节）来开始对交换机局域网的学习。然后仔细学习著名的以太网协议（6.4.2节）。在仔细学习链路层寻址和以太网后，我们将考察链路层交换机的工作方式（6.4.3节），并随后考察通常是如何用这些交换机构建大规模局域网的（6.4.4节）。

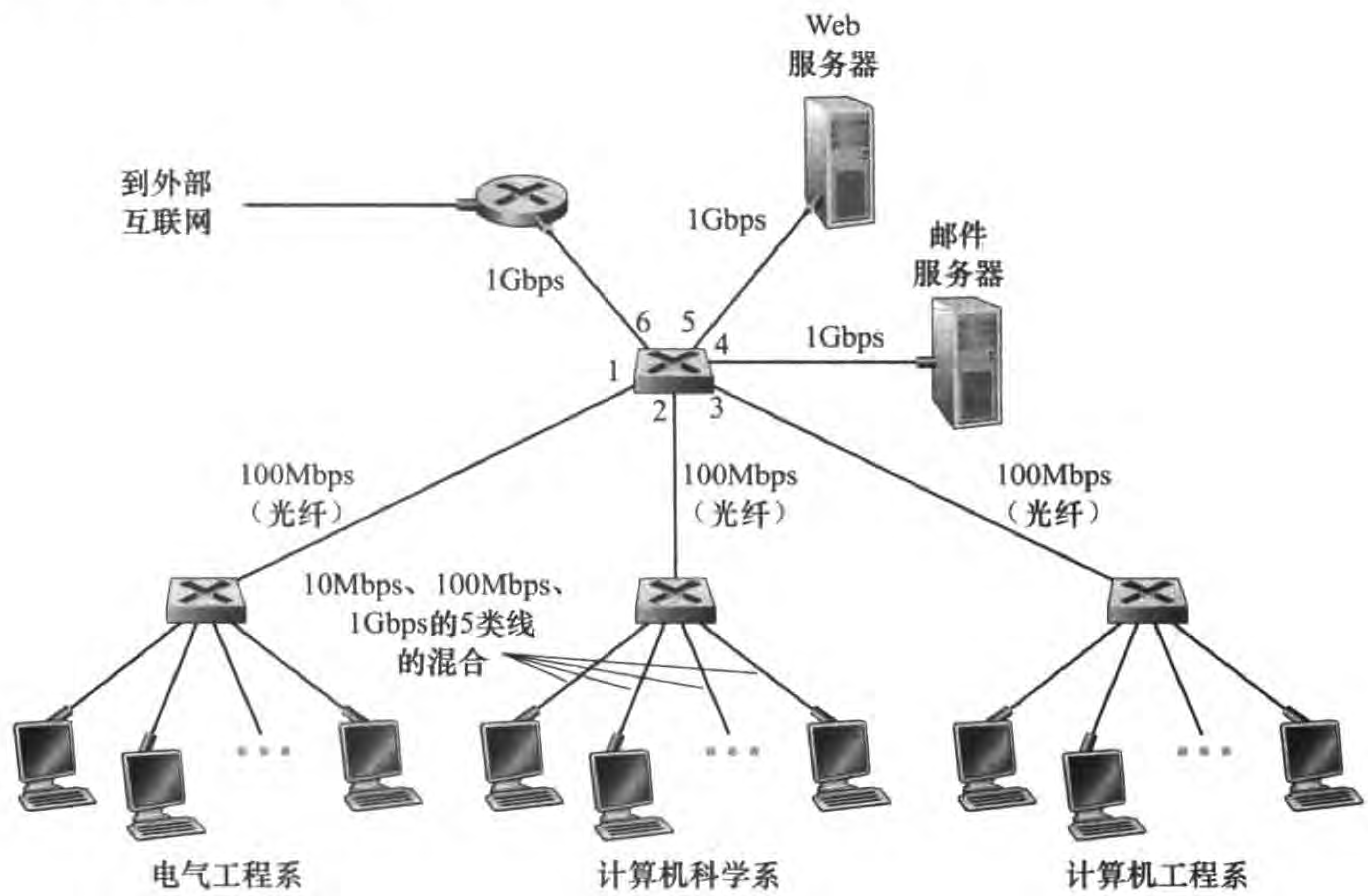


图6-15 由4台交换机连接起来的某机构网络

6.4.1 链路层寻址和ARP

主机和路由器具有链路层地址。现在你也许会感到惊讶，第4章中不是讲过主机和路由器也具有网络层地址吗？你也许会问：为什么我们在网络层和链路层都需要地址呢？除了描述链路层地址的语法和功能，在本节中我们希望明明白白地搞清楚两层地址都有用的原因，事实上这些地址是必不可少的。我们还将学习地址解析协议（ARP），该协议提供了将IP地址转换为链路层地址的机制。

1. MAC地址

事实上，并不是主机或路由器具有链路层地址，而是它们的适配器（即网络接口）具有链路层地址。因此，具有多个网络接口的主机或路由器将具有与之相关联的多个链路层地址，就像它也具有与之相关联的多个IP地址一样。然而，重要的是注意到链路层交换机并不具有与它们的接口（这些接口是与主机和路由器相连的）相关联的链路层地址。这是因为链路层交换机的任务是在主机与路由器之间承载数据报；交换机透明地执行该项任务，这就是说，主机或路由器不必明确地将帧寻址到其间的交换机。图6-16中说明了这

种情况。链路层地址有各种不同的称呼：**LAN 地址**（LAN address）、**物理地址**（physical address）或 **MAC 地址**（MAC address）。因为 MAC 地址似乎是最为流行的术语，所以我们此后就将链路层地址称为 MAC 地址。对于大多数局域网（包括以太网和 802.11 无线局域网）而言，MAC 地址长度为6 字节，共有 2^{48} 个可能的 MAC 地址。如图 6-16 所示，这些6个字节地址通常用十六进制表示法，地址的每个字节被表示为一对十六进制数。尽管 MAC 地址被设计为永久的，但用软件改变一块适配器的 MAC 地址现在是可能的。然而，对于本节的后面部分而言，我们将假设某适配器的 MAC 地址是固定的。

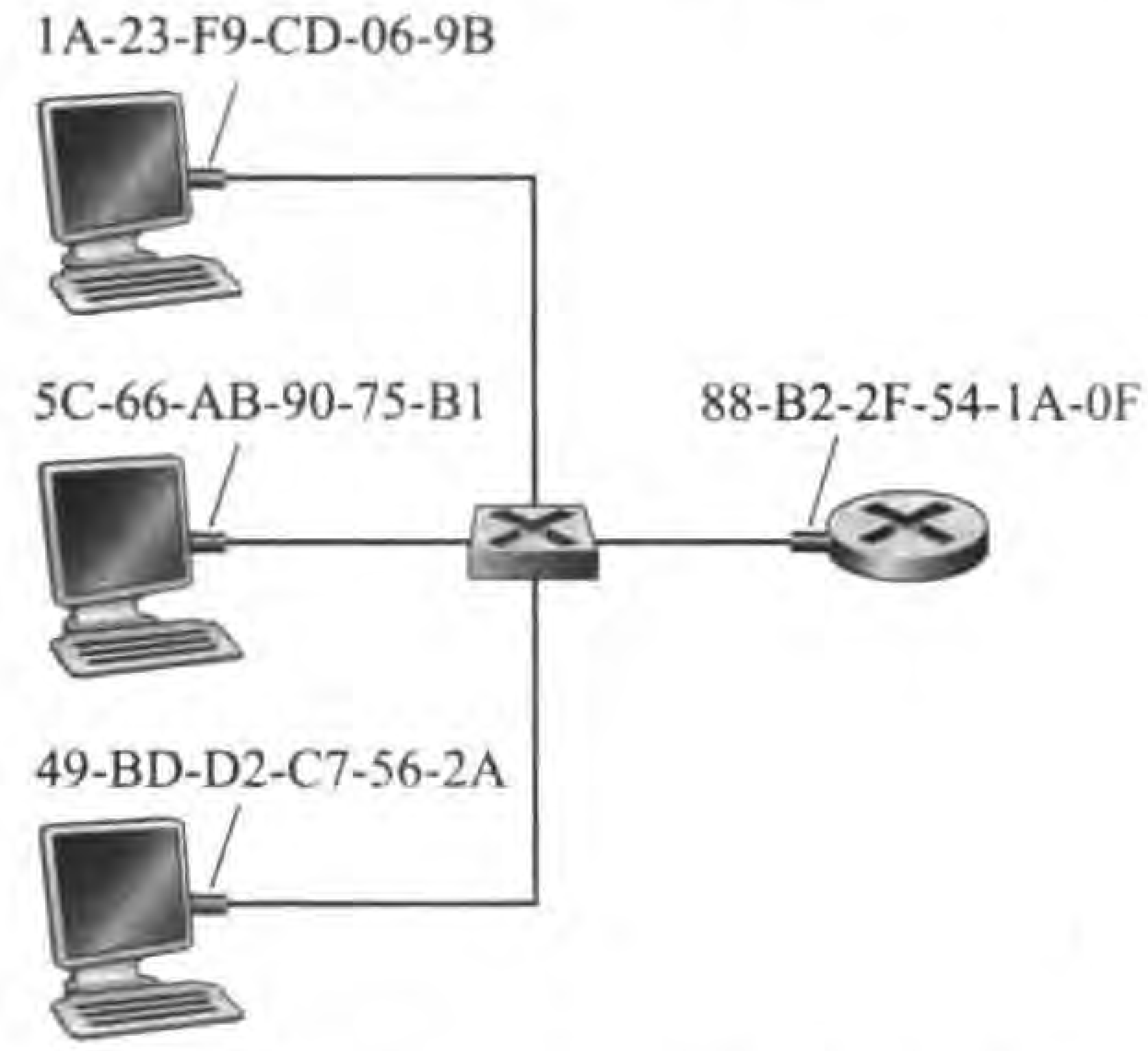


图 6-16　与局域网相连的每个接口都有一个唯一的 MAC 地址

MAC 地址的一个有趣性质是没有两块适配器具有相同的地址。考虑到适配器是由许多不同国家和地区的不同公司生产的，这看起来似乎是件神奇之事。中国台湾生产适配器的公司如何能够保证与比利时生产适配器的公司使用不同的地址呢？答案是 IEEE 在管理着该 MAC 地址空间。特别是，当一个公司要生产适配器时，它支付象征性的费用购买组成 2^{24} 个地址的一块地址空间。IEEE 分配这块 2^{24} 个地址的方式是：固定一个 MAC 地址的前 24 比特，让公司自己为每个适配器生成后 24 比特的唯一组合。

适配器的 MAC 地址具有扁平结构（这与层次结构相反），而且不论适配器到哪里用都不会变化。带有以太网接口的便携机总具有同样的 MAC 地址，无论该计算机位于何方。具有 802.11 接口的一台智能手机总是具有相同的 MAC 地址，无论该智能手机到哪里。与之形成对照的是，前面说过的 IP 地址具有层次结构（即一个网络部分和一个主机部分），而且当主机移动时，主机的 IP 地址需要改变，即改变它所连接到的网络。适配器的 MAC 地址与人的社会保险号相似，后者也具有扁平寻址结构，而且无论人到哪里该号码都不会变化。IP 地址则与一个人的邮政地址相似，它是有层次的，无论何时当人搬家时，该地址都必须改变。就像一个人可能发现邮政地址和社会保险号都有用那样，一台主机具有一个网络层地址和一个 MAC 地址是有用的。

当某适配器要向某些目的适配器发送一个帧时，发送适配器将目的适配器的 MAC 地址插入到该帧中，并将该帧发送到局域网上。如我们马上要看到的那样，一台交换机偶尔将一个入帧广播到它的所有接口。我们将在第 7 章中看到 802.11 也广播帧。因此一块适配器可以接收一个并非向它寻址的帧。这样，当适配器接收到一个帧时，将检查该帧中的目的 MAC 地址是否与它自己的 MAC 地址匹配。如果匹配，该适配器提取出封装的数据报，并将该数据报沿协议栈向上传递。如果不匹配，该适配器丢弃该帧，而不会向上传递该网络层数据报。所以，仅当收到该帧时，才会中断目的地。

然而，有时某发送适配器的确要让局域网上所有其他适配器来接收并处理它打算发送的帧。在这种情况下，发送适配器在该帧的目的地址字段中插入一个特殊的 MAC **广播地址**（broadcast address）。对于使用6 字节地址的局域网（例如以太网和 802.11）来说，广播地址是 48 个连续的 1 组成的字符串（即以十六进制表示法表示的 FF-FF-FF-FF-FF-FF）。

实践原则

保持各层独立

主机和路由器接口除了网络层地址之外还有MAC地址，这有如下几个原因。首先，局域网是为任意网络层协议而设计的，而不只是用于IP和因特网。如果适配器被指派IP地址而不是“中性的”MAC地址的话，则适配器将不能够方便地支持其他网络层协议（例如，IPX或者DECnet）。其次，如果适配器使用网络层地址而不是MAC地址的话，网络层地址必须存储在适配器的RAM中，并且在每次适配器移动（或加电）时要重新配置。另一种选择是在适配器中不使用任何地址，让每个适配器将它收到的每帧数据（通常是IP数据报）沿协议栈向上传递。然后网络层则能够核对网络地址层是否匹配。这种选择带来的一个问题是，主机将被局域网上发送的每个帧中断，包括被目的地是在相同广播局域网上的其他节点的帧中断。总之，为了使网络体系结构中各层次成为极为独立的构建模块，不同的层次需要有它们自己的寻址方案。我们现在已经看到3种类型的地址：应用层的主机名、网络层的IP地址以及链路层的MAC地址。

2. 地址解析协议

因为存在网络层地址（例如，因特网的IP地址）和链路层地址（即MAC地址），所以需要在它们之间进行转换。对于因特网而言，这是**地址解析协议**（Address Resolution Protocol，ARP）[RFC 826]的任务。

为了理解对于诸如ARP这样协议的需求，考虑如图6-17所示的网络。在这个简单的例子中，每台主机和路由器有一个单一的IP地址和单一的MAC地址。与以往一样，IP地址以点分十进制表示法表示，MAC地址以十六进制表示法表示。为了便于讨论，我们在本节中将假设交换机广播所有帧；这就是说，无论何时交换机在一个接口接收一个帧，它将在其所有其他接口上转发该帧。在下一节中，我们将更为准确地解释交换机操作的过程。

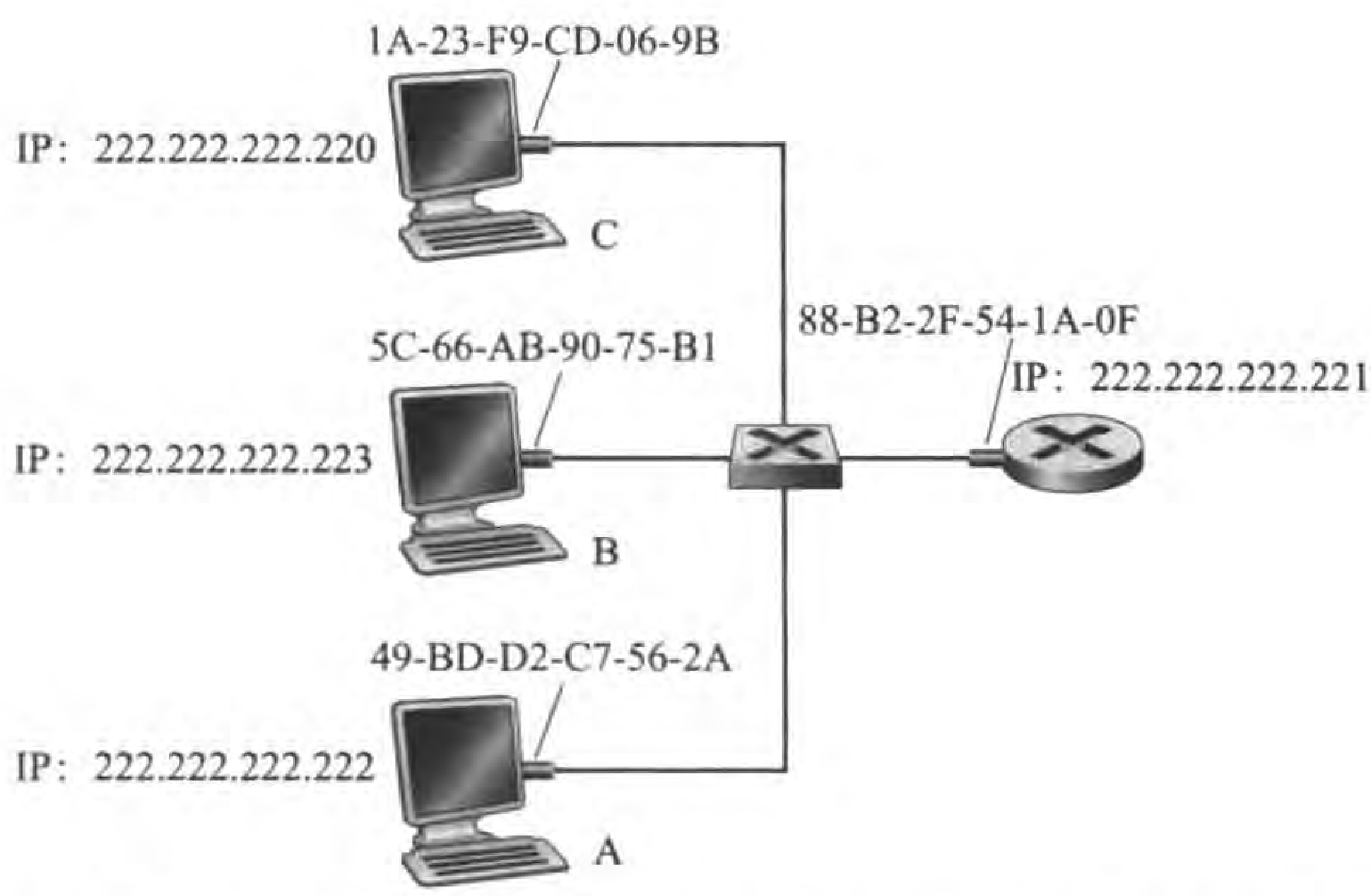


图6-17 局域网上的每个接口都有一个IP地址和一个MAC地址

现在假设IP地址为222.222.222.220的主机要向主机222.222.222.222发送IP数据

报。在本例中，源和目的均位于相同的子网中（在4.3.3节中的寻址意义下）。为了发送数据报，该源必须要向它的适配器不仅提供IP数据报，而且要提供目的主机222.222.222.222的MAC地址。然后发送适配器将构造一个包含目的地的MAC地址的链路层帧，并把该帧发送进局域网。

在本节中要处理的重要问题是，发送主机如何确定IP地址为222.222.222.222的目的主机的MAC地址呢？正如你也许已经猜想的那样，它使用ARP。在发送主机中的ARP模块将取在相同局域网上的任何IP地址作为输入，然后返回相应的MAC地址。在眼下的这个例子中，发送主机222.222.222.220向它的ARP模块提供了IP地址222.222.222.222，并且其ARP模块返回了相应的MAC地址49-BD-D2-C7-56-2A。

因此我们看到了ARP将一个IP地址解析为一个MAC地址。在很多方面它和DNS（在2.4节中学习过）类似，DNS将主机名解析为IP地址。然而，这两种解析器之间的一个重要区别是，DNS为在因特网中任何地方的主机解析主机名，而ARP只为在同一个子网上的主机和路由器接口解析IP地址。如果美国加利福尼亚州的一个节点试图用ARP为美国密西西比州的一个节点解析IP地址，ARP将返回一个错误。

既然已经解释了ARP的用途，我们再来看看它是如何工作的。每台主机或路由器在其内存中具有一个**ARP表**（ARP table），这张表包含IP地址到MAC地址的映射关系。图6-18显示了在主机222.222.222.220中可能看到的ARP表中的内容。该ARP表也包含一个寿命（TTL）值，它指示了从表中删除每个映射的时间。注意到这张表不必为该子网上的每台主机和路由器都包含一个表项；某些可能从来没有进入到该表中，某些可能已经过期。从一个表项放置到某ARP表中开始，一个表项通常的过期时间是20分钟。

IP 地址	MAC 地址	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

图6-18 在主机222.222.222.220中的一个可能的ARP表

现在假设主机222.222.222.220要发送一个数据报，该数据报要IP寻址到本子网上另一台主机或路由器。发送主机需要获得给定IP地址的目的主机的MAC地址。如果发送方的ARP表具有该目的节点的表项，这个任务是很容易完成的。但如果ARP表中当前没有该目的主机的表项，又该怎么办呢？特别是假设222.222.222.220要向222.222.222.222发送数据报。在这种情况下，发送方用ARP协议来解析这个地址。首先，发送方构造一个称为**ARP分组**（ARP packet）的特殊分组。一个ARP分组有几个字段，包括发送和接收IP地址及MAC地址。ARP查询分组和响应分组都具有相同的格式。ARP查询分组的目的是询问子网上所有其他主机和路由器，以确定对应于要解析的IP地址的那个MAC地址。

回到我们的例子上来，222.222.222.220向它的适配器传递一个ARP查询分组，并且指示适配器应该用MAC广播地址（即FF-FF-FF-FF-FF-FF）来发送这个分组。适配器在链路层帧中封装这个ARP分组，用广播地址作为帧的目的地址，并将该帧传输进子网中。回想我们的社会保险号/邮政地址的类比，一次ARP查询等价于一个人在某公司（比方说AnyCorp）一个拥挤的房间大喊："邮政地址是加利福尼亚州帕罗奥图市AnyCorp公司112房间13室的那个人的社会保险号是什么？"包含该ARP查询的帧能被子网上的所有其他

适配器接收到，并且（由于广播地址）每个适配器都把在该帧中的 ARP 分组向上传递给 ARP 模块。这些 ARP 模块中的每个都检查它的 IP 地址是否与 ARP 分组中的目的 IP 地址相匹配。与之匹配的一个给查询主机发送回一个带有所希望映射的响应 ARP 分组。然后查询主机 222.222.222.220 能够更新它的 ARP 表，并发送它的 IP 数据报，该数据报封装在一个链路层帧中，并且该帧的目的 MAC 就是对先前 ARP 请求进行响应的主机或路由器的 MAC 地址。

关于 ARP 协议有两件有趣的事情需要注意。首先，查询 ARP 报文是在广播帧中发送的，而响应 ARP 报文在一个标准帧中发送。在继续阅读之前，你应该思考一下为什么这样。其次，ARP 是即插即用的，这就是说，一个 ARP 表是自动建立的，即它不需要系统管理员来配置。并且如果某主机与子网断开连接，它的表项最终会从留在子网中的节点的表中删除掉。

学生们常常想知道 ARP 是一个链路层协议还是一个网络层协议。如我们所看到的那样，一个 ARP 分组封装在链路层帧中，因而在体系结构上位于链路层之上。然而，一个 ARP 分组具有包含链路层地址的字段，因而可认为是链路层协议，但它也包含网络层地址，因而也可认为是为网络层协议。所以，可能最好把 ARP 看成是跨越链路层和网络层边界两边的协议，即不完全符合我们在第 1 章中学习的简单的分层协议栈。现实世界协议就是这样复杂！

3. 发送数据报到子网以外

现在应该搞清楚当一台主机要向相同子网上的另一台主机发送一个数据报时 ARP 的操作过程。但是现在我们来看更复杂的情况，即当子网中的某主机要向子网之外（也就是跨越路由器的另一个子网）的主机发送网络层数据报的情况。我们在图 6-19 的环境中来讨论这个问题，该图显示了一个由一台路由器互联两个子网所组成的简单网络。

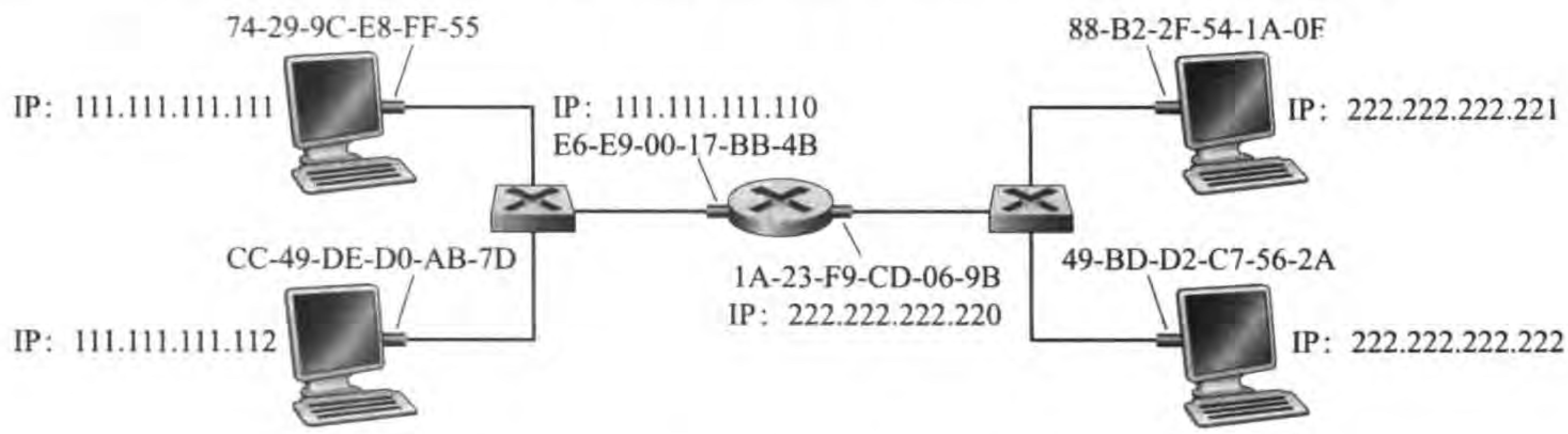


图 6-19 由一台路由器互联的两个子网

有关图 6-19 需要注意几件有趣的事情。每台主机仅有一个 IP 地址和一个适配器。但是，如第 4 章所讨论，一台路由器对它的每个接口都有一个 IP 地址。对路由器的每个接口，（在路由器中）也有一个 ARP 模块和一个适配器。在图 6-19 中的路由器有两个接口，所以它有两个 IP 地址、两个 ARP 模块和两个适配器。当然，网络中的每个适配器都有自己的 MAC 地址。

还要注意到子网 1 的网络地址为 111.111.111/24，子网 2 的网络地址为 222.222.222/24。因此，与子网 1 相连的所有接口都有格式为 111.111.111.xxx 的地址，与子网 2 相连的所有接口都有格式为 222.222.222.xxx 的地址。

现在我们考察子网 1 上的一台主机将向子网 2 上的一台主机发送数据报。特别是，假

设主机 111.111.111.111 要向主机 222.222.222.222 发送一个 IP 数据报。和往常一样，发送主机向它的适配器传递数据报。但是，发送主机还必须向它的适配器指示一个适当的目的 MAC 地址。该适配器应该使用什么 MAC 地址呢？有人也许大胆猜测，这个适当的 MAC 地址就是主机 222.222.222.222 的适配器地址，即 49-BD-D2-C7-56-2A。然而，这个猜测是错误的！如果发送适配器要用那个 MAC 地址，那么子网 1 上所有的适配器都不会费心将该 IP 数据报传递到它的网络层，因为该帧的目的地址与子网 1 上所有适配器的 MAC 地址都将不匹配。这个数据报将只有死亡，到达数据报天国。

如果我们仔细地观察图 6-19，我们发现为了使一个数据报从 111.111.111.111 到子网 2 上的主机，该数据报必须首先发送给路由器接口 111.111.111.110，它是通往最终目的地路径上的第一跳路由器的 IP 地址。因此，对于该帧来说，适当的 MAC 地址是路由器接口 111.111.111.110 的适配器地址，即 E6-E9-00-17-BB-4B。但发送主机怎样获得 111.111.111.110 的 MAC 地址呢？当然是通过使用 ARP！一旦发送适配器有了这个 MAC 地址，它创建一个帧（包含了寻址到 222.222.222.222 的数据报），并把该帧发送到子网 1 中。在子网 1 上的路由器适配器看到该链路层帧是向它寻址的，因此把这个帧传递给路由器的网络层。万岁！该 IP 数据报终于被成功地从源主机移动到这台路由器了！但是我们的任务还没有结束。我们仍然要将该数据报从路由器移动到目的地。路由器现在必须决定该数据报要被转发的正确接口。如在第 4 章中所讨论的，这是通过查询路由器中的转发表来完成的。转发表告诉这台路由器该数据报要通过路由器接口 222.222.222.220 转发。然后该接口把这个数据报传递给它的适配器，适配器把该数据报封装到一个新的帧中，并且将帧发送进子网 2 中。这时，该帧的目的 MAC 地址确实是最终目的地 MAC 地址。路由器又是怎样获得这个目的地 MAC 地址的呢？当然是用 ARP 获得的！

用于以太网的 ARP 定义在 RFC 826 中。在 TCP/IP 指南 RFC 1180 中对 ARP 进行了很好的介绍。我们将在课后习题中更为详细地研究 ARP。

6.4.2 以太网

以太网几乎占领着现有的有线局域网市场。在 20 世纪 80 年代和 90 年代早期，以太网面临着来自其他局域网技术包括令牌环、FDDI 和 ATM 的挑战。多年来，这些其他技术中的一些成功地抓住了部分局域网市场份额。但是自从 20 世纪 70 年代中期发明以太网以来，它就不断演化和发展，并保持了它的支配地位。今天，以太网是到目前为止最流行的有线局域网技术，而且到可能预见的将来它可能仍保持这一位置。可以这么说，以太网对本地区域联网的重要性就像因特网对全球联网所具有的地位那样。

以太网的成功有很多原因。首先，以太网是第一个广泛部署的高速局域网。因为它部署得早，网络管理员非常熟悉以太网（它的奇迹和它的奇思妙想），并当其他局域网技术问世时，他们不愿意转而用之。其次，令牌环、FDDI 和 ATM 比以太网更加复杂、更加昂贵，这就进一步阻碍了网络管理员改用其他技术。第三，改用其他局域网技术（例如 FDDI 和 ATM）的最引人注目的原因通常是这些新技术具有更高数据速率；然而以太网总是奋起抗争，产生了运行在相同或更高数据速率下的版本。20 世纪 90 年代初期引入了交换以太网，这就进一步增加了它的有效数据速率。最后，由于以太网已经很流行了，所以以太网硬件（尤其是适配器和交换机）成了一个普通商品，而且极为便宜。

Bob Metcalfe 和 David Boggs 在 20 世纪 70 年代中期发明初始的以太局域网。初始的以太局域网使用同轴电缆总线来互联节点。以太网的总线拓扑实际上从 20 世纪 80 年代到 90

年代中期一直保持不变。使用总线拓扑的以太网是一种广播局域网，即所有传输的帧传送到与该总线连接的所有适配器并被其处理。回忆一下，我们在 6.3.2 节中讨论了以太网的具有二进制指数回退的 CSMA/CD 多路访问协议。

到了 20 世纪 90 年代后期，大多数公司和大学使用一种基于集线器的星形拓扑以太网安装替代了它们的局域网。在这种安装中，主机（和路由器）直接用双绞对铜线与一台集线器相连。**集线器**（hub）是一种物理层设备，它作用于各个比特而不是作用于帧。当表示一个 0 或一个 1 的比特到达一个接口时，集线器只是重新生成这个比特，将其能量强度放大，并将该比特向其他所有接口传输出去。因此，采用基于集线器的星形拓扑的以太网也是一个广播局域网，即无论何时集线器从它的一个接口接收到一个比特，它向其所有其他接口发送该比特的副本。特别是，如果某集线器同时从两个不同的接口接收到帧，将出现一次碰撞，生成该帧的节点必须重新传输该帧。

在 21 世纪初，以太网又经历了一次重要的革命性变化。以太网安装继续使用星形拓扑，但是位于中心的集线器被**交换机**（switch）所替代。在本章后面我们将深入学习交换以太网。眼下我们仅知道交换机不仅是"无碰撞的"，而且也是名副其实的存储转发分组交换机就可以了；但是与运行在高至第三层的路由器不同，交换机仅运行在第二层。

1. 以太网帧结构

以太网帧如图 6-20 所示。通过仔细研究以太网的帧，我们能够学到许多有关以太网的知识。



图 6-20 以太网帧结构

为了将对以太网帧的讨论放到切实的环境中，考虑从一台主机向另一台主机发送一个 IP 数据报，且这两台主机在相同的以太局域网上（例如，如图 6-17 所示的以太局域网）。（尽管以太网帧的负载是一个 IP 数据报，但我们注意到以太网帧也能够承载其他网络层分组。）设发送适配器（即适配器 A）的 MAC 地址是 AA-AA-AA-AA-AA-AA，接收适配器（即适配器 B）的 MAC 地址是 BB-BB-BB-BB-BB-BB。发送适配器在一个以太网帧中封装了一个 IP 数据报，并把该帧传递到物理层。接收适配器从物理层收到这个帧，提取出 IP 数据报，并将该 IP 数据报传递给网络层。我们现在在这种情况下考察如图 6-20 所示的以太网帧的 6 个字段：

- 数据字段（46～1500 字节）。这个字段承载了 IP 数据报。以太网的最大传输单元（MTU）是 1500 字节。这意味着如果 IP 数据报超过了 1500 字节，则主机必须将该数据报分片，如 4.3.2 节所讨论。数据字段的最小长度是 46 字节。这意味着如果 IP 数据报小于 46 字节，数据报必须被填充到 46 字节。当采用填充时，传递到网络层的数据包括 IP 数据报和填充部分。网络层使用 IP 数据报首部中的长度字段来去除填充部分。
- 目的地址（6 字节）。这个字段包含目的适配器的 MAC 地址，即 BB-BB-BB-BB-BB-BB。当适配器 B 收到一个以太网帧，帧的目的地址无论是 BB-BB-BB-BB-BB-BB，还是 MAC 广播地址，它都将该帧的数据字段的内容传递给网络层；如果它收到了具有任何其他 MAC 地址的帧，则丢弃之。

- 源地址（6字节）。这个字段包含了传输该帧到局域网上的适配器的MAC地址，在本例中为AA-AA-AA-AA-AA-AA。
- 类型字段（2字节）。类型字段允许以太网复用多种网络层协议。为了理解这点，我们需要记住主机能够使用除了IP以外的其他网络层协议。事实上，一台给定的主机可以支持多种网络层协议，以对不同的应用采用不同的协议。因此，当以太网帧到达适配器B，适配器B需要知道它应该将数据字段的内容传递给哪个网络层协议（即分解）。IP和其他链路层协议（例如，Novell IPX或AppleTalk）都有它们各自的、标准化的类型编号。此外，ARP协议（在上一节讨论过）有自己的类型编号，并且如果到达的帧包含ARP分组（即类型字段的值为十六进制的0806），则该ARP分组将被多路分解给ARP协议。注意到该类型字段和网络层数据报中的协议字段、运输层报文段的端口号字段相类似；所有这些字段都是为了把一层中的某协议与上一层的某协议结合起来。
- CRC（4字节）。如6.2.3节中讨论的那样，CRC（循环冗余检测）字段的目的是使得接收适配器（适配器B）检测帧中是否引入了差错。
- 前同步码（8字节）。以太网帧以一个8字节的前同步码（Preamble）字段开始。该前同步码的前7字节的值都是10101010；最后一个字节是10101011。前同步码字段的前7字节用于"唤醒"接收适配器，并且将它们的时钟和发送方的时钟同步。为什么这些时钟会不同步呢？记住适配器A的目的是根据以太局域网类型的不同，分别以10Mbps、100Mbps或者1Gbps的速率传输帧。然而，没有什么是完美无缺的，因此适配器A不会以精确的额定速率传输帧；相对于额定速率总有一些漂移，局域网上的其他适配器不会预先知道这种漂移的。接收适配器只需通过锁定前同步码的前7字节的比特，就能够锁定适配器A的时钟。前同步码的第8个字节的最后两个比特（第一个出现的两个连续的1）警告适配器B，"重要的内容"就要到来了。

所有的以太网技术都向网络层提供无连接服务。这就是说，当适配器A要向适配器B发送一个数据报时，适配器A在一个以太网帧中封装该数据报，并且把该帧发送到局域网上，没有先与适配器B握手。这种第二层的无连接服务类似于IP的第三层数据报服务和UDP的第四层无连接服务。

以太网技术都向网络层提供不可靠服务。特别是，当适配器B收到一个来自适配器A的帧，它对该帧执行CRC校验，但是当该帧通过CRC校验时它既不发送确认帧；而当该帧没有通过CRC校验时它也不发送否定确认帧。当某帧没有通过CRC校验，适配器B只是丢弃该帧。因此，适配器A根本不知道它传输的帧是否到达了B并通过了CRC校验。（在链路层）缺乏可靠的传输有助于使得以太网简单和便宜。但是它也意味着传递到网络层的数据报流能够有间隙。

如果由于丢弃了以太网帧而存在间隙，主机B上的应用也会看见这个间隙吗？如我们在第3章中学习的那样，这只取决于该应用是使用UDP还是使用TCP。如果应用使用的是UDP，则主机B中的应用的确会看到数据中的间隙。另一方面，如果应用使用的是TCP，则主机B中的TCP将不会确认包含在丢弃帧中的数据，从而引起主机A的TCP重传。注意到当TCP重传数据时，数据最终将回到曾经丢弃它的以太网适配器。因此，从这种意义上来说，以太网的确重传了数据，尽管以太网并不知道它是正在传输一个具有全新数据的全新数据报，还是一个包含已经被传输过至少一次的数据的数据报。

历史事件

Bob Metcalfe 和以太网

作为20世纪70年代早期哈佛大学的一名博士生，Bob Metcalfe 在 MIT 从事 ARPAnet 的研究。在他学习期间，他还受到了 Abramson 有关 ALOHA 和随机接入协议工作的影响。在完成了他的博士学位，并在开始 Xerox Palo Alto 研究中心（Xerox PARC）的工作之前，他用3个月访问了 Abramson 和他在夏威夷大学的同事，获得了 ALOHAnet 的第一手资料。在 Xerox PARC，Metcalfe 受到了 Alto 计算机的影响，这种计算机在很多方面是20世纪80年代个人计算机的先驱。Metcalfe 看到了对这些计算机以一种不昂贵的方式组网的需求。因此，基于他在 APRAnet、ALOHAnet 和随机接入协议方面的知识，Metcalfe 和他的同事 David Boggs 一起发明了以太网。

Metcalfe 和 Boggs 的初始以太网运行速度为 2.94Mbps，连接长达一英里范围的多达256台主机。Metcalfe 和 Boggs 成功地使得 Xerox PARC 的大多数研究人员通过他们的 Alto 计算机互相通信。然后 Metcalfe 推进了 Xerox、Digital 和 Intel 联盟，创建了以太网作为一种 10Mbps 的以太网标准，该标准后被 IEEE 认可。Xerox 对以太网商业化没有表现出太多的兴趣。1979年，Metcalfe 建立了自己的公司 3Com，它发展和商业化包括以太网技术在内的联网技术。特别是，3Com 在20世纪80年代早期为非常流行的 IBM PC 开发了以太网网卡并使之市场化。

2. 以太网技术

在以上的讨论中我们已经提到以太网，仿佛它有单一的协议标准似的。但事实上，以太网具有许多不同的特色，具有某种令人眼花缭乱的首字母缩写词，如 10BASE-T、10BASE-2、100BASE-T、1000BASE-LX 和 10GBASE-T。这些以及许多其他的以太网技术在多年中已经被 IEEE 802.3 CSMA/CD(Ethernet) 工作组标准化了［IEEE 802.3 2012］。尽管这些首字母缩写词看起来眼花缭乱，实际上其中非常有规律性。首字母缩写词的第一部分指该标准的速率：10、100、1000 或 10G，分别代表 10Mbps、100Mbps、1000Mbps（或 1Gbps）和 10Gbps 以太网。“BASE”指基带以太网，这意味着该物理媒体仅承载以太网流量；几乎所有的 802.3 标准都适用于基带以太网。该首字母缩写词的最后一部分指物理媒体本身；以太网是链路层也是物理层的规范，并且能够经各种物理媒体（包括同轴电缆、铜线和光纤）承载。一般而言，“T”指双绞铜线。

从历史上讲，以太网最初被构想为一段同轴电缆。早期的 10BASE-2 和 10BASE-5 标准规定了在两种类型的同轴电缆之上的 10Mbps 以太网，每种标准都限制在500米长度之内。通过使用**转发器**（repeater）能够得到更长的运行距离，而转发器是一种物理层设备，它能在输入端接收信号并在输出端再生该信号。同轴电缆很好地对应于我们将作为一种广播媒体的以太网视图，即由一个接口传输的所有帧可在其他接口收到，并且以太网的 CSMA/CD 协议很好地解决了多路访问问题。节点直接附着在电缆上，万事大吉，我们有了一个局域网了！

多年来以太网已经经历了一系列演化步骤，今天的以太网非常不同于使用同轴电缆的初始总线拓扑的设计。在今天大多数的安装中，节点经点对点的由双绞铜线或光纤线缆构成的线段与一台交换机相连，如图6-15至图6-17所示。

在20世纪90年代中期，以太网被标准化为100Mbps，比10Mbps以太网快10倍。初始的以太网MAC协议和帧格式保留了下来，但更高速率的物理层被定义为用铜线（100BASE-T）和用光纤（100BASE-FX、100BASE-SX、100BASE-BX）。图6-21显示了这些不同的标准和共同的以太网MAC协议和帧格式。100Mbps以太网用双绞线距离限制为100米，用光纤距离限制为几千米，允许把不同建筑物中的以太网交换机连接起来。

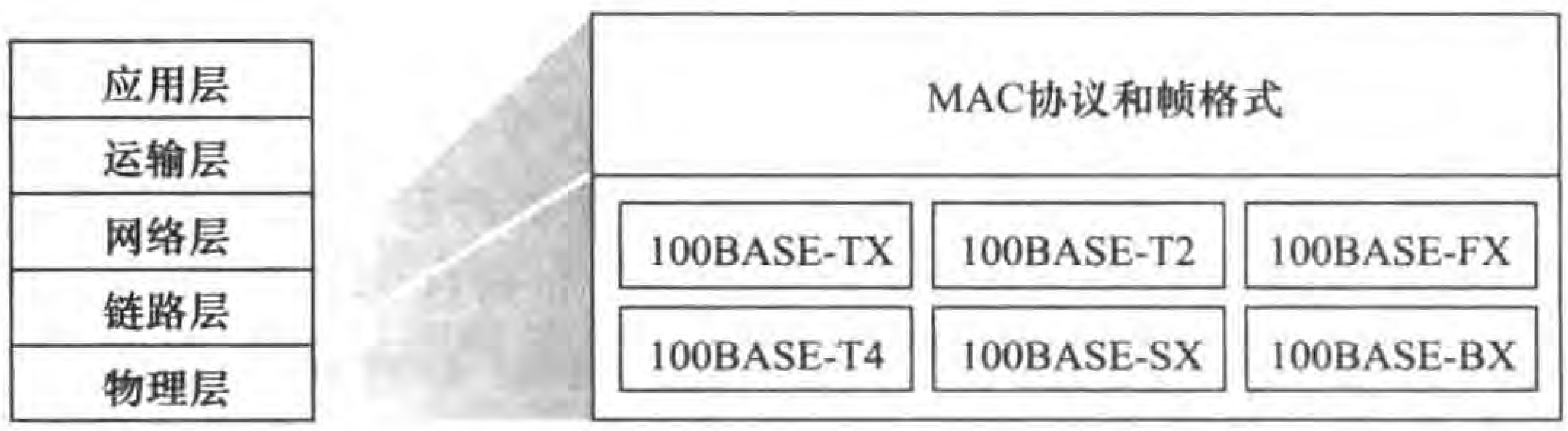


图6-21　100Mbps以太网标准：共同的链路层，不同的物理层

吉比特以太网是对极为成功的10Mbps和100Mbps以太网标准的扩展。40Gbps以太网提供40 000Mbps的总数据速率，与大量已经安装的以太网设备基础保持完全兼容。吉比特以太网的标准称为IEEE 802.3z，它完成以下工作：

- 使用标准以太网帧格式（参见图6-20），并且后向兼容10BASE-T与100BASE-T技术。这使得吉比特以太网和现已安装的以太网设备基础很容易集成。
- 允许点对点链路以及共享的广播信道。如前所述，点对点链路使用交换机，而广播信道使用集线器。在吉比特以太网术语中，集线器被称为“带缓存的分配器”。
- 使用CSMA/CD来共享广播信道。为了得到可接受的效率，节点之间的最大距离必须严格限制。
- 对于点对点信道，允许在两个方向上都以40Gbps全双工操作。

吉比特以太网最初工作于光纤之上，现在能够工作在5类UTP线缆上。

我们通过提出一个问题来结束有关以太网技术的讨论，这个问题开始可能会难倒你。在总线拓扑和基于集线器的星形拓扑技术时代，以太网很显然是一种广播链路（如6.3节所定义），其中多个节点同时传输时会出现帧碰撞。为了处理这些碰撞，以太网标准包括了CSMA/CD协议，该协议对于跨越一个小的地理半径的有线广播局域网特别有效。但是对于今天广为使用的以太网是基于交换机的星形拓扑，采用的是存储转发分组交换，是否还真正需要一种以太网MAC协议呢？如我们很快所见，交换机协调其传输，在任何时候决不会向相同的接口转发超过一个帧。此外，现代交换机是全双工的，这使得一台交换机和一个节点能够在同时向对方发送帧而没有干扰。换句话说，在基于交换机的以太局域网中，不会有碰撞，因此没有必要使用MAC协议了！

如我们所见，今天的以太网与Metcalfe和Boggs在30多年前构想的初始以太网有非常大的不同，即速度已经增加了3个数量级，以太网帧承载在各种各样的媒体之上，交换以太网已经成为主流，此时甚至连MAC协议也经常是不必要的了！所有这些还真正是以太网吗？答案当然是：“是的，根据定义如此。”然而，注意到下列事实是有趣的：通过所有这些改变，的确还有一个历经30年保持未变的持久不变量，即以太网帧格式。也许这才是以太网标准的一个真正重要的特征。

6.4.3　链路层交换机

到目前为止，我们有意对交换机实际要做的工作以及它是怎样工作的含糊其辞。交换

机的任务是接收入链路层帧并将它们转发到出链路；我们将在这一节中详细学习这种转发功能。我们将看到交换机自身对子网中的主机和路由器是**透明的**（transparent）；这就是说，某主机/路由器向另一个主机/路由器寻址一个帧（而不是向交换机寻址该帧），顺利地将该帧发送进局域网，并不知道某交换机将会接收该帧并将它转发到另一个节点。这些帧到达该交换机的任何输出接口之一的速率可能暂时会超过该接口的链路容量。为了解决这个问题，交换机输出接口设有缓存，这非常类似于路由器接口为数据报设有缓存。现在我们来仔细考察交换机运行的原理。

1. 交换机转发和过滤

过滤（filtering）是决定一个帧应该转发到某个接口还是应当将其丢弃的交换机功能。**转发**（forwarding）是决定一个帧应该被导向哪个接口，并把该帧移动到那些接口的交换机功能。交换机的过滤和转发借助于**交换机表**（switch table）完成。该交换机表包含某局域网上某些主机和路由器的但不必是全部的表项。交换机表中的一个表项包含：①一个MAC地址；②通向该MAC地址的交换机接口；③表项放置在表中的时间。图6-22中显示了图6-15中最上方交换机的一个交换机表的例子。尽管帧转发的描述听起来类似于第4章讨论的数据转发，但我们将很快看到它们之间有重要的差异。确实在4.4节中一般化转发的讨论中我们学习过，许多现代分组交换机能够被配置，以基于第二层目的MAC地址（即起着第二层交换机的功能）或者第三层IP目的地址（即起着第三层交换机的功能）进行转发。无论如何，我们将对交换机基于MAC地址而不是基于IP地址转发分组进行明确区分。我们也将看到传统的（即处于非SDN环境）交换机表的构造方式与路由器转发表的构造方式有很大不同。

地址	接口	时间
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
…	…	…

图6-22　图6-15中最上面交换机的交换机表的一部分

为了理解交换机过滤和转发的工作过程，假定目的地址为DD-DD-DD-DD-DD-DD的帧从交换机接口 x 到达。交换机用MAC地址DD-DD-DD-DD-DD-DD索引它的表。有3种可能的情况：

- 表中没有对于DD-DD-DD-DD-DD-DD的表项。在这种情况下，交换机向除接口 x 外的所有接口前面的输出缓存转发该帧的副本。换言之，如果没有对于目的地址的表项，交换机广播该帧。
- 表中有一个表项将DD-DD-DD-DD-DD-DD与接口 x 联系起来。在这种情况下，该帧从包括适配器DD-DD-DD-DD-DD-DD的局域网网段到来。无须将该帧转发到任何其他接口，交换机通过丢弃该帧执行过滤功能即可。
- 表中有一个表项将DD-DD-DD-DD-DD-DD与接口 $y \neq x$ 联系起来。在这种情况下，该帧需要被转发到与接口 y 相连的局域网网段。交换机通过将该帧放到接口 y 前面的输出缓存完成转发功能。

我们大致地看一下用于图6-15中最上面交换机的这些规则以及图6-22中所示的它的交换机表。假设目的地址为62-FE-F7-11-89-A3的一个帧从接口1到达该交换机。交换机检查它的表并且发现其目的地是在与接口1相连的局域网网段上（即电气工程系的局域

网)。这意味着该帧已经在包含目的地的局域网网段广播过了。因此该交换机过滤（即丢弃）了该帧。现在假设有同样目的地址的帧从接口 2 到达。交换机再次检查它的表并且发现其目的地址在接口 1 的方向上；因此它向接口 1 前面的输出缓存转发该帧。这个例子清楚地表明，只要交换机的表是完整和准确的，该交换机无须任何广播就向着目的地转发帧。

在这种意义上，交换机比集线器更为“聪明”。但是一开始这个交换机表是如何配置起来的呢？链路层有与网络层路由选择协议等价的协议吗？或者必须要一名超负荷工作的管理员人工地配置交换机表吗？

2. 自学习

交换机具有令人惊奇的特性（特别是对于早已超负荷工作的网络管理员），那就是它的表是自动、动态和自治地建立的，即没有来自网络管理员或来自配置协议的任何干预。换句话说，交换机是**自学习**（self-learning）的。这种能力是以如下方式实现的：

1）交换机表初始为空。

2）对于在每个接口接收到的每个入帧，该交换机在其表中存储：①在该帧源地址字段中的 MAC 地址；②该帧到达的接口；③当前时间。交换机以这种方式在它的表中记录了发送节点所在的局域网网段。如果在局域网上的每个主机最终都发送了一个帧，则每个主机最终将在这张表中留有记录。

3）如果在一段时间（称为老化期（aging time））后，交换机没有接收到以该地址作为源地址的帧，就在表中删除这个地址。以这种方式，如果一台 PC 被另一台 PC（具有不同的适配器）代替，原来 PC 的 MAC 地址将最终从该交换机表中被清除掉。

我们粗略地看一下用于图 6-15 中最上面交换机的自学习性质以及在图 6-22 中它对应的交换机表。假设在时刻 9:39，源地址为 01-12-23-34-45-56 的一个帧从接口 2 到达。假设这个地址不在交换机表中。于是交换机在其表中增加一个新的表项，如图 6-23 中所示。

地址	接口	时间
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
…	…	…

图 6-23　交换机学习到地址为 01-12-23-34-45-56 的适配器所在的位置

继续这个例子，假设该交换机的老化期是 60min，在 9:32 ~ 10:32 期间源地址是 62-FE-F7-11-89-A3 的帧没有到达该交换机。那么在时刻 10:32，这台交换机将从它的表中删除该地址。

交换机是**即插即用设备**（plug-and-play device），因为它们不需要网络管理员或用户的干预。要安装交换机的网络管理员除了将局域网网段与交换机的接口相连外，不需要做其他任何事。管理员在安装交换机或者当某主机从局域网网段之一被去除时，他没有必要配置交换机表。交换机也是双工的，这意味着任何交换机接口能够同时发送和接收。

3. 链路层交换机的性质

在描述了链路层交换机的基本操作之后，我们现在来考虑交换机的特色和性质。我们能够指出使用交换机的几个优点，它们不同于如总线或基于集线器的星形拓扑那样的广播链路：

- 消除碰撞。在使用交换机（不使用集线器）构建的局域网中，没有因碰撞而浪费

的带宽！交换机缓存帧并且决不会在网段上同时传输多于一个帧。就像使用路由器一样，交换机的最大聚合带宽是该交换机所有接口速率之和。因此，交换机提供了比使用广播链路的局域网高得多的性能改善。

- 异质的链路。交换机将链路彼此隔离，因此局域网中的不同链路能够以不同的速率运行并且能够在不同的媒体上运行。例如，图 6-22 中最上面的交换机有 3 条 1Gbps 1000BASE-T 铜缆链路、2 条 100Mbps 10BASE-FX 光缆链路和 1 条 100 BASE-T 铜缆链路。因此，对于原有的设备与新设备混用，交换机是理想的。
- 管理。除了提供强化的安全性（参见插入材料“关注安全性”），交换机也易于进行网络管理。例如，如果一个适配器工作异常并持续发送以太网帧（称为快而含糊的（jabbering）适配器），交换机能够检测到该问题，并在内部断开异常适配器。有了这种特色，网络管理员不用起床并开车到工作场所去解决这个问题。类似地，一条割断的缆线仅使得使用该条缆线连接到交换机的主机断开连接。在使用同轴电缆的时代，许多网络管理员花费几个小时“沿线巡检”（或者更准确地说“在天花板上爬行”），以找到使整个网络瘫痪的电缆断开之处。交换机也收集带宽使用的统计数据、碰撞率和流量类型，并使这些信息为网络管理者使用。这些信息能够用于调试和解决问题，并规划该局域网在未来应当演化的方式。研究人员还在原型系统部署中探讨在以太局域网中增加更多的管理功能［Casado 2007；Koponen 2011］。

关注安全性

嗅探交换局域网：交换机毒化

当一台主机与某交换机相连时，它通常仅接收到明确发送给它的帧。例如，考虑在图 6-17 中的一个交换局域网。当主机 A 向主机 B 发送帧时，在交换机表中有用于主机 B 的表项，则该交换机将仅向主机 B 转发该帧。如果主机 C 恰好在运行嗅探器，主机 C 将不能够嗅探到 A 到 B 的帧。因此，在交换局域网的环境中（与如 802.11 局域网或基于集线器的以太局域网的广播链路环境形成对比），攻击者嗅探帧更为困难。然而，因为交换机广播那些目的地址不在交换机表中的帧，位于 C 上的嗅探器仍然能嗅探某些不是明确寻址到 C 的帧。此外，嗅探器将能够嗅探到具有广播地址 FF-FF-FF-FF-FF-FF 的广播帧。一个众所周知的对抗交换机的攻击称为**交换机毒化**（switch poisoning），它向交换机发送大量的具有不同伪造源 MAC 地址的分组，因而用伪造表项填满了交换机表，没有为合法主机留下空间。这使该交换机广播大多数帧，这些帧则能够由嗅探器俘获到［Skoudis 2006］。由于这种攻击只有技艺高超的攻击者才能做到，因此交换机比起集线器和无线局域网来更难受到嗅探。

4. 交换机和路由器比较

如我们在第 4 章学习的那样，路由器是使用网络层地址转发分组的存储转发分组交换机。尽管交换机也是一个存储转发分组交换机，但它和路由器是根本不同的，因为它用 MAC 地址转发分组。交换机是第二层的分组交换机，而路由器是第三层的分组交换机。然而，回顾我们在 4.4 节中所学习的内容，使用“匹配加动作”的现代交换机能够转发基

于帧的目的 MAC 地址的第二层帧，也能转发使用数据报目的 IP 地址的第三层数据报。我们的确看到了使用 OpenFlow 标准的交换机能够基于 11 个不同的帧、数据报和运输层首部字段，执行通用的分组转发。

即使交换机和路由器从根本上是不同的，网络管理员在安装互联设备时也经常必须在它们之间进行选择。例如，对于图 6-15 中的网络，网络管理员本来可以很容易地使用路由器而不是交换机来互联各个系的局域网、服务器和互联网网关路由器。路由器的确使得各系之间通信而不产生碰撞。既然交换机和路由器都是候选的互联设备，那么这两种方式的优点和缺点各是什么呢?

首先考虑交换机的优点和缺点。如上面提到的那样，交换机是即插即用的，这是世界上所有超负荷工作的网络管理员都喜爱的特性。交换机还能够具有相对高的分组过滤和转发速率，就像图 6-24 中所示的那样，交换机必须处理高至第二层的帧，而路由器必须处理高至第三层的数据报。在另一方面，为了防止广播帧的循环，交换网络的活跃拓扑限制为一棵生成树。另外，一个大型交换网络将要求在主机和路由器中有大的 ARP 表，这将生成可观的 ARP 流量和处理量。而且，交换机对于广播风暴并不提供任何保护措施，即如果某主机出了故障并传输出没完没了的以太网广播帧流，该交换机将转发所有这些帧，使得整个以太网的崩溃。

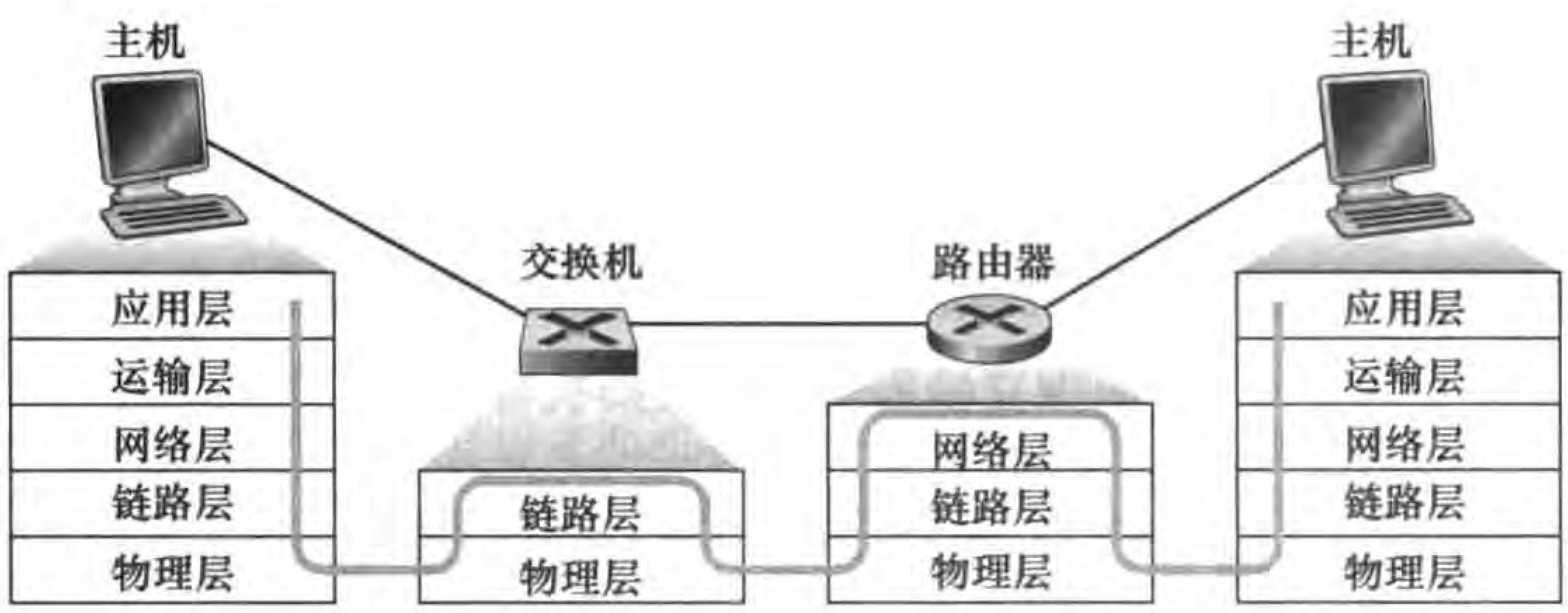


图 6-24 在交换机、路由器和主机中分组的处理

现在考虑路由器的优点和缺点。因为网络寻址通常是分层次的（不像 MAC 寻址那样是扁平的)，即使当网络中存在冗余路径时，分组通常也不会通过路由器循环。(然而，当路由器表被误配置时，分组可能循环；但是如我们在第 4 章所知，IP 用一个特殊的报文首部字段来限制循环。）所以，分组就不会被限制到一棵生成树上，并可以使用源和目的地之间的最佳路径。因为路由器没有生成树限制，所以它们允许以丰富的拓扑结构构建因特网，例如包括欧洲和北美之间的多条活跃链路。路由器的另一个特色是它们对第二层的广播风暴提供了防火墙保护。尽管也许路由器最重要的缺点就是它们不是即插即用的，即路由器和连接到它们的主机都需要人为地配置 IP 地址。而且路由器对每个分组的处理时间通常比交换机更长，因为它们必须处理高达第三层的字段。最后，路由器一词有两种不同的发音方法，或者发音为“rootor”或发音为“rowter”，人们浪费了许多时间争论正确的发音［Perlman 1999］。

给出了交换机和路由器各自具有的优点和缺点后（总结在表 6-1 中），一个机构的网络（例如，大学校园网或者公司园区网）什么时候应该使用交换机，什么时候应该使

表 6-1 流行的互联设备的典型特色的比较

	集线器	路由器	交换机
流量隔离	无	有	有
即插即用	有	无	有
优化路由	无	有	无

用路由器呢？通常，由几百台主机组成的小网络通常有几个局域网网段。对于这些小网络，交换机就足够了，因为它们不要求 IP 地址的任何配置就能使流量局部化并增加总计吞吐量。但是在由几千台主机组成的更大网络中，通常在网络中（除了交换机之外）还包括路由器。路由器提供了更健壮的流量隔离方式和对广播风暴的控制，并在网络的主机之间使用更"智能的"路由。

对于交换网络和路由网络的优缺点的进一步讨论，以及如何能够将交换局域网技术扩展为比今天的以太网容纳多两个数量级以上的主机，参见［Meyers 2004；Kim 2008］。

6.4.4 虚拟局域网

在前面图 6-15 的讨论中，我们注意到现代机构的局域网常常是配置为等级结构的，每个工作组（部门）有自己的交换局域网，经过一个交换机等级结构与其他工作组的交换局域网互联。虽然这样的配置在理想世界中能够很好地工作，但在现实世界常常不尽如人意。在图 6-15 中的配置中，能够发现 3 个缺点：

- *缺乏流量隔离*。尽管该等级结构把组流量局域化到一个单一交换机中，但广播流量（例如携带 ARP 和 DHCP 报文或那些目的地还没有被自学习交换机学习到的帧）仍然必须跨越整个机构网络。限制这些广播流量的范围将改善局域网的性能。也许更为重要的是，为了安全/隐私的目的也可能希望限制局域网广播流量。例如，如果一个组包括公司的行政管理团队，另一个组包括运行着 Wireshark 分组嗅探器的心怀不满的雇员，网络管理员也许非常希望行政流量无法到达该雇员的主机。通过用路由器代替图 6-15 中的中心交换机，能够提供这种类型的隔离。我们很快看到这种隔离也能够经过一种交换（第二层）解决方案来取得。
- *交换机的无效使用*。如果该机构不止有 3 个组，而是有 10 个组，则将要求有 10 个第一级交换机。如果每个组都较小，比如说少于 10 个人，则单台 96 端口的交换机将足以容纳每个人，但这台单一的交换机将不能提供流量隔离。
- *管理用户*。如果一个雇员在不同组间移动，必须改变物理布线，以将该雇员连接到图 6-15 中的不同的交换机上。属于两个组的雇员将使问题更为困难。

幸运的是，这些难题中的每个都能够通过支持**虚拟局域网**（Virtula Local Network，VLAN）的交换机来处理。顾名思义，支持 VLAN 的交换机允许经一个单一的物理局域网基础设施定义多个虚拟局域网。在一个 VLAN 内的主机彼此通信，仿佛它们（并且没有其他主机）与交换机连接。在一个基于端口的 VLAN 中，交换机的端口（接口）由网络管理员划分为组。每个组构成一个 VLAN，在每个 VLAN 中的端口形成一个广播域（即来自一个端口的广播流量仅能到达该组中的其他端口）。图 6-25 显示了具有 16 个端口的单一交换机。端口 2～8 属于电气工程系（EE）VLAN，而端口 9～15 属于计算机科学系（CS）VLAN（端口 1 和 16 未分配）。这个 VLAN 解决了上面提到的所有困难，即 EE VLAN 帧和 CS VLAN 帧彼此隔离，图 6-15 中的两台交换机已由一台交换机替代，并且在交换机端口 8 的用户加入计算机科学系时，网络操作员只需重新配置 VLAN 软件，

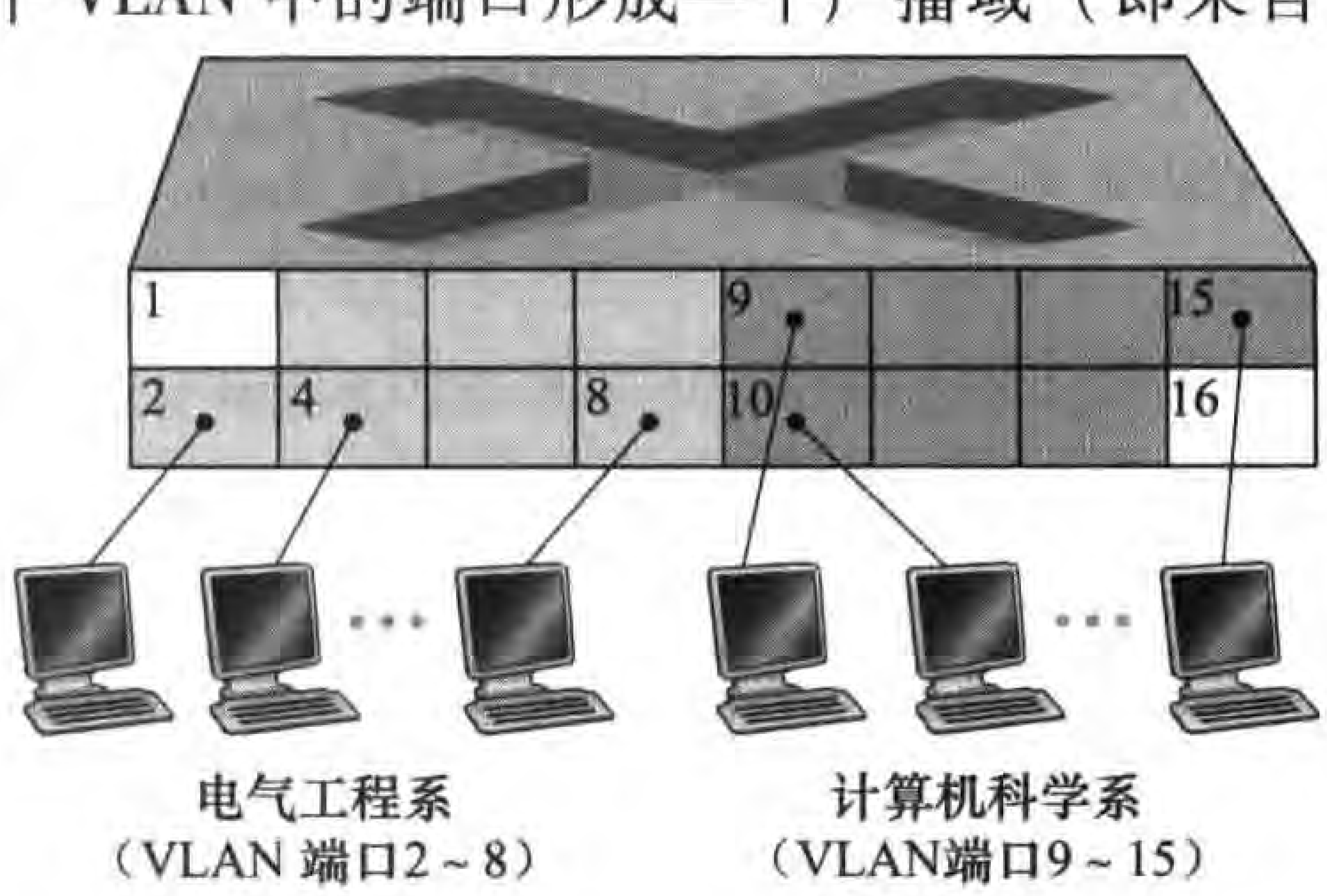


图 6-25 配置了两个 VLAN 的单台交换机

使得端口8与CS VLAN相关联即可。人们容易想象到VLAN交换机配置和操作的方法，即网络管理员使用交换机管理软件声明一个端口属于某个给定的VLAN（其中未声明的端口属于一个默认的VLAN)，在交换机中维护一张端口到VLAN的映射表；交换机软件仅在属于相同VLAN的端口之间交付帧。

但完全隔离两个VLAN带来了新的困难！来自电子工程系的流量怎样才能发送到计算机科学系呢？解决这个问题的一种方式是将VLAN交换机的一个端口（例如在图6-25中的端口1）与一台外部的路由器相连，并且将该端口配置为属于EE VLAN和CS VLAN。在此情况下，即使电子工程系和计算机科学系共享相同的物理交换机，其逻辑配置看起来也仿佛是电子工程系和计算机科学系具有分离的经路由器连接的交换机。从电子工程系发往计算机科学系的数据报将首先跨越EE VLAN到达路由器，然后由该路由器转发跨越CS VLAN到达CS主机。幸运的是交换机厂商使这种配置变得容易，网络管理员通过构建包含一台VLAN交换机和一台路由器的单一设备，这样就不再需要分离的外部路由器了。本章后面的课后习题中更为详细地探讨了这种情况。

再次返回到图6-15，我们现在假设计算机工程系没有分离开来，某些电子工程和计算机科学教职员工位于一座建筑物中，他们当然需要网络接入，并且他们希望成为他们系VLAN的一部分。图6-26显示了第二台8端口交换机，其中交换机端口已经根据需要定义为属于EE VLAN或CS VLAN。但是这两台交换机应当如何互联呢？一种容易的解决方案是在每台交换机上定义一个属于CS VALN的端口（对EE VLAN也类似处理)，并且如图6-26a所示将这两个端口彼此互联起来。然而，这种解决方案不具有扩展性，因为在每台交换机上N个VLAN将要求N个端口直接互联这两台交换机。

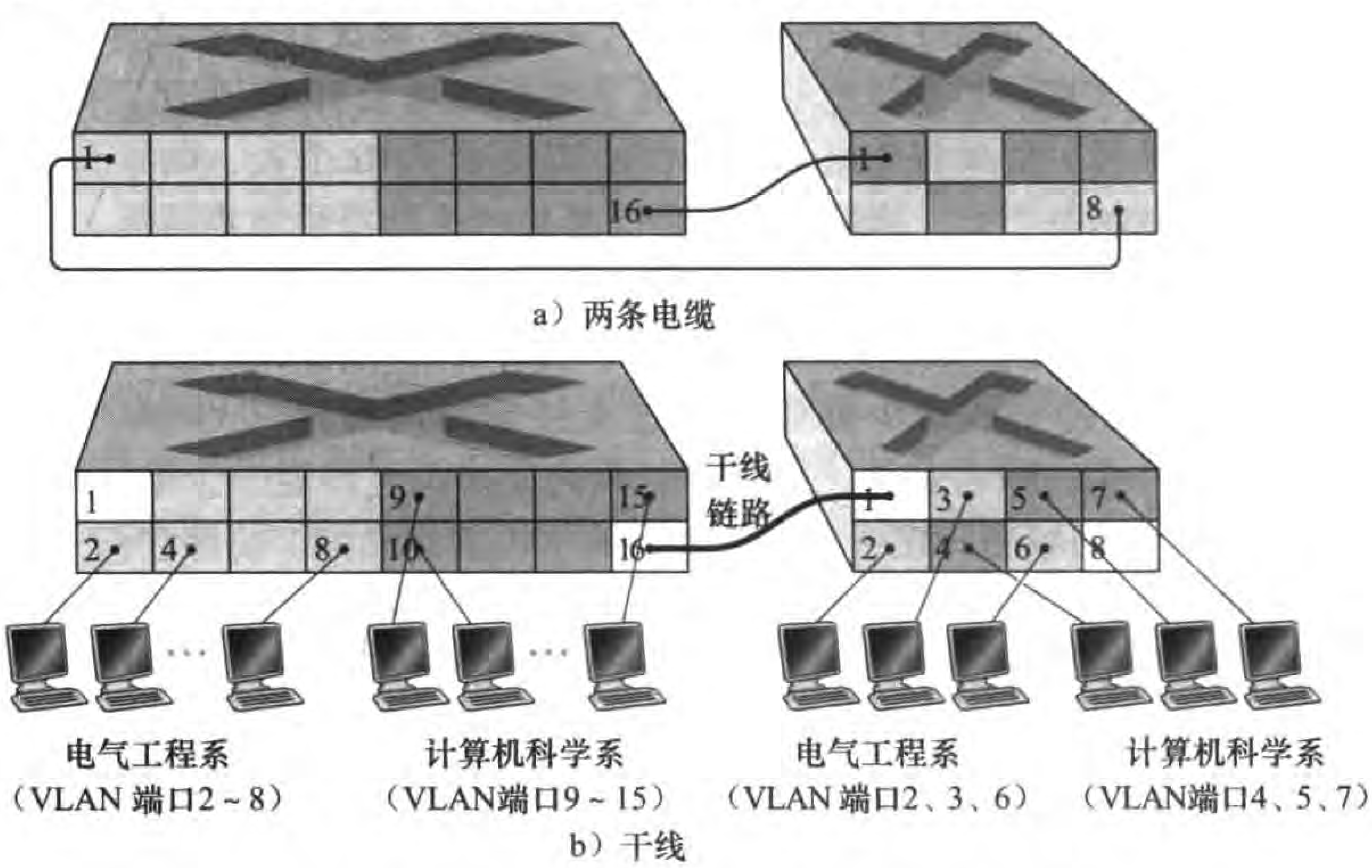


图6-26 连接具有两个VLAN的两台VLAN交换机

一种更具扩展性互联VLAN交换机的方法称为**VLAN干线连接**（VLAN trunking)。在图6-26b所示的VLAN干线方法中，每台交换机上的一个特殊端口（左侧交换机上的端口16，右侧交换机上的端口1）被配置为干线端口，以互联这两台VLAN交换机。该干线端口属于所有VLAN，发送到任何VLAN的帧经过干线链路转发到其他交换机。但这会引起另外的问题：一个交换机怎样知道到达干线端口的帧属于某个特定的VLAN呢？IEEE定

义了一种扩展的以太网帧格式——802.1Q，用于跨越 VLAN 干线的帧。如图 6-27 中所示，802.1Q 帧由标准以太网帧与加进首部的 4 字节 **VLAN 标签**（VLAN tag）组成，而 VLAN 标签承载着该帧所属的 VLAN 标识符。VLAN 标签由在 VLAN 干线发送侧的交换机加进帧中，解析后并由在 VLAN 干线接收侧的交换机删除。VLAN 标签自身由一个 2 字节的**标签协议标识符**（Tag Protocol Identifier，TPID）字段（具有固定的十六进制值 81-00）、一个 2 字节的标签控制信息字段（包含一个 12 比特的 VLAN 标识符字段）和一个 3 比特优先权字段（具有类似于 IP 数据报 TOS 字段的目的）组成。



图 6-27 初始的以太网帧（上部），802.1Q 标签以太网 VLAN 帧（下部）

在这部分讨论中，我们仅仅简要地涉及了 VLAN，关注了基于端口的 VLAN。我们也应当提及 VLAN 能够以几种其他方式定义。在基于 MAC 的 VLAN 中，网络管理员指定属于每个 VLAN 的 MAC 地址的集合；无论何时一个设备与一个端口连接时，端口基于设备的 MAC 地址将其连接进适当的 VLAN。VLAN 也能基于网络层协议（例如 IPv4、IPv6 或 Appletalk）和其他准则进行定义。VLAN 跨越 IP 路由器扩展也是可能的，这使得多个 LAN 孤岛能被连接在一起，以形成能够跨越全局的单一 LAN［Yu 2011］。详情请参见 802.1Q 标准［IEEE 802.1q 2005］。

6.5 链路虚拟化：网络作为链路层

因为本章关注链路层协议，所以在我们临近该章结束的时候，让我们反思一下对已经演化的词汇链路的理解。在本章开始时，我们将链路视为连接两台通信主机的物理线路。在学习多路访问协议时，我们看到了多台主机能够通过一条共享的线路连接起来，并且连接主机的这种“线路”能够是无线电频谱或其他媒体。这使我们将该链路更多地抽象为一条信道，而不是作为一条线路。在我们学习以太局域网时（图 6-15），我们看到互联媒体实际上能够是一种相当复杂的交换基础设施。然而，经过这种演化，主机本身维持着这样的视图，即互联媒体只是连接两台或多台主机的链路层信道。我们看到，例如一台以太网主机不知道它是通过单一短局域网网段（图 6-17）还是通过地理上分布的交换局域网（图 6-15）或通过 VLAN 与其他局域网主机进行连接，这是很幸福的事。

在两台主机之间由拨号调制解调器连接的场合，连接这两台主机的链路实际上是电话网，这是一个逻辑上分离的、全球性的电信网络，它有自己的用于数据传输和信令的交换机、链路和协议栈。然而，从因特网链路层的观点看，通过电话网的拨号连接被看作一根简单的“线路”。在这个意义上，因特网虚拟化了电话网，将电话网看成为两台因特网主机之间提供链路层连接的链路层技术。你可能回想起在第 2 章中对于覆盖网络的讨论，类似地，一个覆盖网络将因特网视为为覆盖节点之间提供连接性的一种手段，寻求以因特网覆盖电话网的相同方式来覆盖因特网。

在本节中，我们将考虑多协议标签交换（MPLS）网络。与电路交换的电话网不同，MPLS 客观上讲是一种分组交换的虚电路网络。它们有自己的分组格式和转发行为。因此，从教学法的观点看，有关 MPLS 的讨论既适合放在网络层的学习中，也适合放在链路层的学习中。然而，从因特网的观点看，我们能够认为 MPLS 像电话网和交换以太网一样，作为为 IP 设备提供互联服务的链路层技术。因此，我们将在链路层讨论中考虑 MPLS。帧中继和 ATM 网络也能用于互联 IP 设备，虽然这些技术看上去有些过时（但仍在部署），这里将不再讨论；详情请参见一本可读性强的书［Goralski 1999］。我们对 MPLS 的讨论将是简明扼要的，因为有关这些网络每个都能够写（并且已经写了）整本书。有关 MPLS 详情我们推荐［Davie 2000］。我们这里主要关注这些网络怎样为互联 IP 设备提供服务，尽管我们也将更深入一些探讨支撑基础技术。

多协议标签交换

多协议标签交换（Multiprotocol Label Switching，MPLS）自 20 世纪 90 年代中后期在一些产业界的努力下进行演化，以改善 IP 路由器的转发速度。它采用来自虚电路网络领域的一个关键概念：固定长度标签。其目标是：对于基于固定长度标签和虚电路的技术，在不放弃基于目的地 IP 数据报转发的基础设施的前提下，当可能时通过选择性地标识数据报并允许路由器基于固定长度的标签（而不是目的地 IP 地址）转发数据报来增强其功能。重要的是，这些技术与 IP 协同工作，使用 IP 寻址和路由选择。IETF 在 MPLS 协议中统一了这些努力［RFC 3031；RFC 3032］，有效地将虚电路（VC）技术综合进了路由选择的数据报网络。

首先考虑由 MPLS 使能的路由器处理的链路层帧格式，以此开始学习 MPLS。图 6-28 显示了在 MPLS 使能的路由器之间传输的一个链路层帧，该帧具有一个小的 MPLS 首部，该首部增加到第二层（如以太网）首部和第三层（即 IP）首部之间。RFC 3032 定义了用于这种链路的 MPLS 首部的格式；用于 ATM 和帧中继网络的首部也定义在其他的 RFC 文档中。包括在 MPLS 首部中的字段是：标签；预留的 3 比特实验字段；1 比特 S 字段，用于指示一系列“成栈”的 MPLS 首部的结束（我们这里不讨论这个高级主题）；寿命字段。

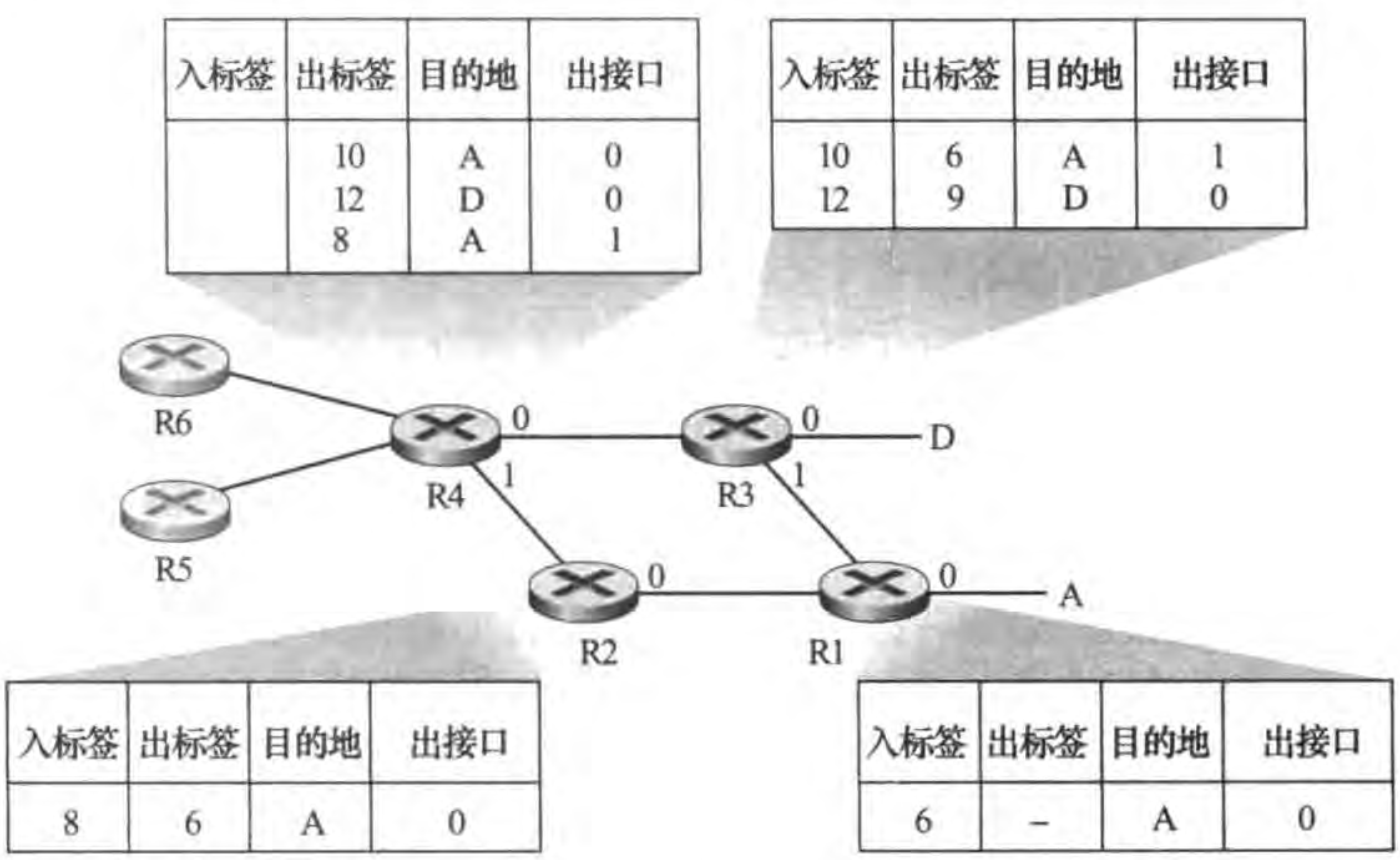


图 6-28 MPLS 首部：位于链路层和网络层首部之间

从图 6-28 立即能够看出，一个 MPLS 加强的帧仅能在两个均为 MPLS 使能的路由器之间发送。（因为一个非 MPLS 使能的路由器，当它在期望发现 IP 首部的地方发现了一个 MPLS 首部时会相当混淆！）一个 MPLS 使能的路由器常被称为**标签交换路由器**（label-switched router），因为它通过在其转发表中查找 MPLS 标签，然后立即将数据报传递给适当的输出接口来转发 MPLS 帧。因此，MPLS 使能的路由器不需要提取目的 IP 地址和在转发表中执行最长前缀匹配的查找。但是路由器怎样才能知道它的邻居是否的确是 MPLS 使能的呢？路由器如何知道哪个标签与给定 IP 目的地相联系呢？为了回答这些问题，我们需要看看一组 MPLS 使能路由器之间的交互过程。

在图 6-29 所示的例子中，路由器 R1 到 R4 都是 MPLS 使能的，R5 和 R6 是标准的 IP

路由器。R1 向 R2 和 R3 通告了它（R1）能够路由到目的地 A，并且具有 MPLS 标签 6 的接收帧将要转发到目的地 A。路由器 R3 已经向路由器 R4 通告了它能够路由到目的地 A 和 D，分别具有 MPLS 标签 10 和 12 的入帧将朝着这些目的地交换。路由器 R2 也向路由器 R4 通告了它（R2）能够到达目的地 A，具有 MPLS 标签 8 的接收帧将朝着 A 交换。注意到路由器 R4 现在处于一个到达 A 且有两个 MPLS 路径的令人感兴趣的位置上，经接口 0 具有出 MPLS 标签 10，经接口 1 具有出 MPLS 标签 8。在图 6-29 中画出的外围部分是 IP 设备 R5、R6、A 和 D，它们经过一个 MPLS 基础设施（MPLS 使能路由器 R1、R2、R3 和 R4）连接在一起，这与一个交换局域网或 ATM 网络能够将 IP 设备连接到一起的方式十分相似。并且与交换局域网或 ATM 网络相似，MPLS 使能路由器 R1 到 R4 完成这些工作时从没有接触分组的 IP 首部。



R4：

入标签	出标签	目的地	出接口
	10	A	0
	12	D	0
	8	A	1

R3：

入标签	出标签	目的地	出接口
10	6	A	1
12	9	D	0

R2：

入标签	出标签	目的地	出接口
8	6	A	0

R1：

入标签	出标签	目的地	出接口
6	–	A	0

图 6-29　MPLS 增强的转发

在我们上面的讨论中，我们并没有指定在 MPLS 使能路由器之间分布标签的特定协议，因为该信令的细节已经超出了本书的范围。然而，我们注意到，IETF 的 MPLS 工作组已经在［RFC 3468］中定义了 RSVP 协议的一种扩展，称之为 RSVP-TE［RFC 3209］，它将关注对 MPLS 信令所做的工作。我们也不讨论 MPLS 实际上是如何计算在 MPLS 使能路由器之间分组的路径的，也不讨论它如何收集链路状态信息（例如，未由 MPLS 预留的链路带宽量）以用于这些路径计算中。现有的链路状态路由选择算法（例如 OSPF）已经扩展为向 MPLS 使能路由器"洪泛"。令人感兴趣的是，实际路径计算算法没有标准化，它们当前是厂商特定的算法。

至今为止，我们关于 MPLS 的讨论重点基于这样的事实，MPLS 基于标签执行交换，而不必考虑分组的 IP 地址。然而，MPLS 的真正优点和当前对 MPLS 感兴趣的原因并不在于交换速度的潜在增加，而在于 MPLS 使能的新的流量管理能力。如前面所述，R4 到 A 具有两条 MPLS 路径。如果转发在 IP 层基于 IP 地址执行，我们在第 4 章中学习的 IP 路由选择协议将只指定到 A 的单一最小费用的路径。所以，MPLS 提供了沿着多条路由转发分组的能力，使用标准 IP 路由选择协议这些路由将是不可能的。这是使用 MPLS 的一种简单形式的**流量工程**（traffic engineering）［RFC 3346；RFC3272；RFC 2702；Xiao 2000］，其中网络运行者能够超越普通的 IP 路由选择，迫使某些流量沿着一条路径朝着某给定的目的

地引导，并且朝着相同目的地的其他流量沿着另一条路径流动（无论是由于策略、性能或某些其他原因）。

将 MPLS 用于其他目的也是可能的。能用于执行 MPLS 转发路径的快速恢复，例如，经过一条预计算的无故障路径重路由流量来对链路故障做出反应［Kar 2000；Huang 2002；RFC 3469］。最后，我们注意到 MPLS 能够并且已经被用于实现所谓**虚拟专用网**（Virtual Private Network，VPN）。在为用户实现一个 VPNR 的过程中，ISP 使用它的 MPLS 使能网络将用户的各种网络连接在一起。MPLS 能被用于将资源和由用户的 VPN 使用的寻址方式相隔离，其他用户利用该 VPN 跨越该 ISP 网络，详情参见［DeClercq 2002］。

这里有关 MPLS 的讨论是简要的，我们鼓励读者查阅我们提到的这些文献。我们注意到对 MPLS 有许多可能的用途，看起来它将迅速成为因特网流量工程的瑞士军刀！

6.6　数据中心网络

近年来，因特网公司如谷歌、微软、脸书（Facebook）和亚马逊（以及它们在亚洲和欧洲的同行）已经构建了大量的数据中心。每个数据中心都容纳了数万至数十万台主机，并且同时支持着很多不同的云应用（例如搜索、电子邮件、社交网络和电子商务）。每个数据中心都有自己的**数据中心网络**（data center network），这些数据中心网络将其内部主机彼此互联并与因特网中的数据中心互联。在本节中，我们简要介绍用于云应用的数据中心网络。

大型数据中心的投资巨大，一个有 100 000 台主机的数据中心每个月的费用超过 1200 万美元［Greenberg 2009a］。在该费用中，用于主机自身的开销占45%（每3~4年需要更新一次）；变压器、不间断电源系统、长时间断电时使用的发电机以及冷却系统等基础设施的开销占25%；用于功耗的电力设施的开销占15%；用于联网的开销占15%，这包括了网络设备（交换机、路由器和负载均衡设备）、外部链路以及传输流量的开销。（在这些比例中，设备费用是分期偿还的，因此费用通常是由一次性购买和持续开销（如能耗）构成的。）尽管联网不是最大的费用，但是网络创新是减少整体成本和性能最大化的关键［Greenberg 2009a］。

主机就像是数据中心的工蜂：它们负责提供内容（例如，网页和视频），存储邮件和文档，并共同执行大规模分布式计算（例如，为搜索引擎提供分布式索引计算）。数据中心中的主机称为**刀片**（blade），与比萨饼盒类似，一般是包括 CPU、内存和磁盘存储的商用主机。主机被堆叠在机架上，每个机架一般堆放 20~40 台刀片。在每一个机架顶部有一台交换机，这台交换机被形象地称为**机架顶部**（Top of Rack，TOR）**交换机**，它们与机架上的主机互联，并与数据中心中的其他交换机互联。具体来说，机架上的每台主机都有一块与 TOR 交换机连接的网卡，每台 TOR 交换机有额外的端口能够与其他 TOR 交换机连接。目前主机通常用 40Gbps 的以太网连接到它们的 TOR 交换机［Greenberg 2015］。每台主机也会分配一个自己的数据中心内部的 IP 地址。

数据中心网络支持两种类型的流量：在外部客户与内部主机之间流动的流量，以及内部主机之间流动的流量。为了处理外部客户与内部主机之间流动的流量，数据中心网络包括了一台或者多台**边界路由器**（border router），它们将数据中心网络与公共因特网相连。数据中心网络因此需要将所有机架彼此互联，并将机架与边界路由器连接。图 6-30 显示了一个数据中心网络的例子。**数据中心网络设计**（data center network design）是互联网络

和协议设计的艺术，该艺术专注于机架彼此连接和与边界路由器相连。近年来，数据中心网络的设计已经成为计算机网络研究的重要分支［Al-Fares 2008；Greenberg 2009a；Greenberg 2009b；Mysore 2009；Guo 2009；Wang 2010］。

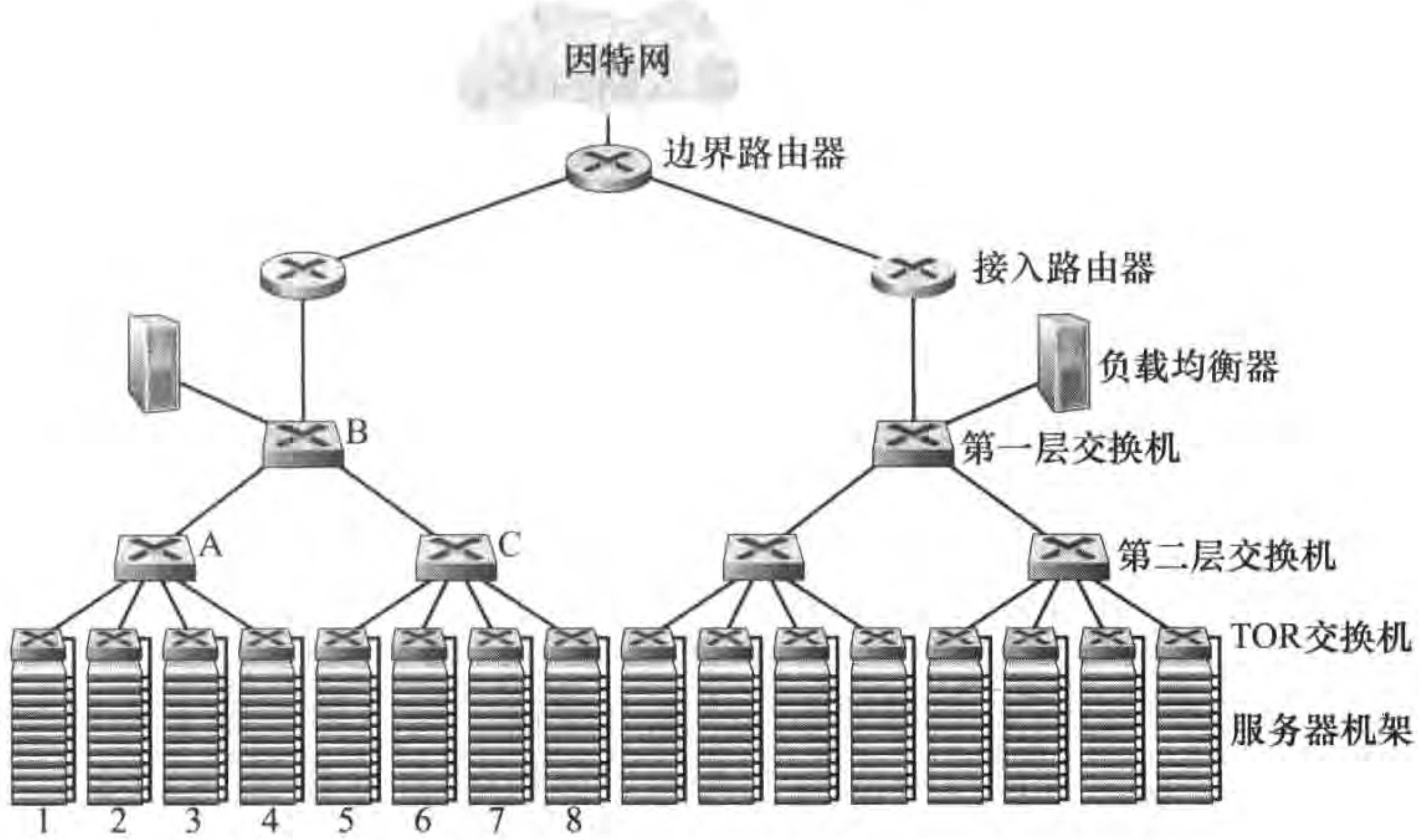


图 6-30 具有等级拓扑的数据中心网络

1. 负载均衡

一个云数据中心，如一个谷歌或者微软的数据中心，能够同时提供诸如搜索、电子邮件和视频应用等许多应用。为了支持来自外部客户的请求，每一个应用都与一个公开可见的 IP 地址关联，外部用户向该地址发送其请求并从该地址接收响应。在数据中心内部，外部请求首先被定向到一个**负载均衡器**（load balancer）。负载均衡器的任务是向主机分发请求，以主机当前的负载作为函数来在主机之间均衡负载。一个大型的数据中心通常会有几台负载均衡器，每台服务于一组特定的云应用。由于负载均衡器基于分组的目的端口号（第四层）以及目的 IP 地址做决策，因此它们常被称为“第四层交换机”。一旦接收到一个对于特定应用程序的请求，负载均衡器将该请求分发到处理该应用的某一台主机上（该主机可能再调用其他主机的服务来协助处理该请求）。当主机处理完该请求后，向负载均衡器回送响应，再由负载均衡器将其中继发回给外部客户。负载均衡器不仅平衡主机间的工作负载，而且还提供类似 NAT 的功能，将外部 IP 地址转换为内部适当主机的 IP 地址，然后将反方向流向客户的分组按照相反的转换进行处理。这防止客户直接接触主机，从而具有隐藏网络内部结构和防止客户直接与主机交互等安全性益处。

2. 等级体系结构

对于仅有数千台主机的小型数据中心，一个简单的网络也许就足够了。这种简单网络由一台边界路由器、一台负载均衡器和几十个机架组成，这些机架由单一以太网交换机进行互联。但是当主机规模扩展到几万至几十万的时候，数据中心通常应用**路由器和交换机等级结构**（hierarchy of router and switch），图 6-30 显示了这样的拓扑。在该等级结构的顶端，边界路由器与接入路由器相连（在图 6-30 中仅仅显示了两台，但是能够有更多）。在每台接入路由器下面，有 3 层交换机。每台接入路由器与一台第一层交换机相连，每台第一层交换机与多台第二层交换机以及一台负载均衡器相连。每台第二层交换机又通过机架的 TOR 交换机（第三层交换机）与多个机架相连。所有链路通常使用以太网作为链路层

和物理层协议，并混合使用铜缆和光缆。通过这种等级式设计，可以将数据中心扩展到几十万台主机的规模。

因为云应用提供商持续地提供高可用性的应用是至关重要的，所以数据中心在它们的设计中也包含了冗余网络设备和冗余链路（在图6-30中没有显示出来）。例如，每台TOR交换机能够与两台第二层交换机相连，每台接入路由器、第一层交换机和第二层交换机可以冗余并集成到设计中［Cisco 2012；Greenberg 2009b］。在图6-30中的等级设计可以看到，每台接入路由器下的这些主机构成了单一子网。为了使ARP广播流量本地化，这些子网的每个都被进一步划分为更小的VLAN子网，每个由数百台主机组成［Greenberg 2009a］。

尽管刚才描述的传统的等级体系结构解决了扩展性问题，但是依然存在*主机到主机容量受限*的问题［Greenberg 2009b］。为了理解这种限制，重新考虑图6-30，并且假设每台主机用1Gbps链路连接到它的TOR交换机，而交换机间的链路是10Gbps的以太网链路。在相同机架中的两台主机总是能够以1Gbps全速通信，而只受限于主机网络接口卡的速率。然而，如果在数据中心网络中同时存在多条并发流，则不同机架上的两台主机间的最大速率会小得多。为了深入理解这个问题，考虑不同机架上的40对不同主机间的40条并发流的情况。具体来说，假设图6-30中机架1上10台主机都向机架5上对应的主机发送一条流。类似地，在机架2和机架6的主机对上有10条并发流，机架3和机架7间有10条并发流，机架4和机架8间也有10条并发流。如果每一条流和其他流经同一条链路的流平均地共享链路容量，则经过10Gbps的A到B链路（以及10Gbps的B到C链路）的40条流中每条流获得的速率为10Gbps/40＝250Mbps，显著小于1Gbps的网络接口卡速率。如果主机间的流量需要穿过该等级结构的更高层，这个问题会变得更加严重。对这个限制的一种可行的解决方案是部署更高速率的交换机和路由器。但是这会大大增加数据中心的费用，因为具有高接口速率的交换机和路由器是非常昂贵的。

因为数据中心的一个关键需求是放置计算和服务的灵活性，所以支持主机到主机的高带宽通信十分重要［Greenberg 200b；Farrington 2010］。例如，一个大规模的因特网搜索引擎可能运行在跨越多个机架的上千台主机上，在所有主机对之间具有极高的带宽要求。类似地，像EC2这样的云计算服务可能希望将构成用户服务的多台虚拟机运行在具有最大容量的物理主机上，而无须考虑它们在数据中心的位置。如果这些物理主机跨越了多个机架，前面描述的网络瓶颈可能会导致性能不佳。

3. 数据中心网络的发展趋势

为了降低数据中心的费用，同时提高其在时延和吞吐量上的性能，因特网云服务巨头如谷歌、脸书、亚马逊和微软都在不断地部署新的数据中心网络设计方案。尽管这些设计方案都是专有的，但是许多重要的趋势是一样的。

其中的一个趋势是部署能够克服传统等级设计缺陷的新型互联体系结构和网络协议。一种方法是采用**全连接拓扑**（fully connected topology）来替代交换机和路由器的等级结构［Facebook 2014；Al-Fares 2008；Greenberg 2009b；Guo 2009］，图6-31中显示了这种拓扑。在这种设计中，每台第一层交换机都与所有第二层交换机相连，因此：①主机到主机的流量绝不会超过该交换机层次；②对于n台第一层交换机，在任意两台二层交换机间有n条不相交的路径。这种设计可以显著地改善主机到主机的容量。为了理解该问题，重新考虑40条流的例子。图6-31中的拓扑能够处理这种流模式，因为在第1台第二层交换机

和第2台第二层交换机间存在4条不相交的路径，可以一起为前两台第二层交换机之间提供总和为40Gbps的聚合容量。这种设计不仅减轻了主机到主机的容量限制，同时创建了一种更加灵活的计算和服务环境。在这种环境中，任何未连接到同一台交换机的两个机架之间的通信在逻辑上是等价的，而不论其在数据中心的位置如何。

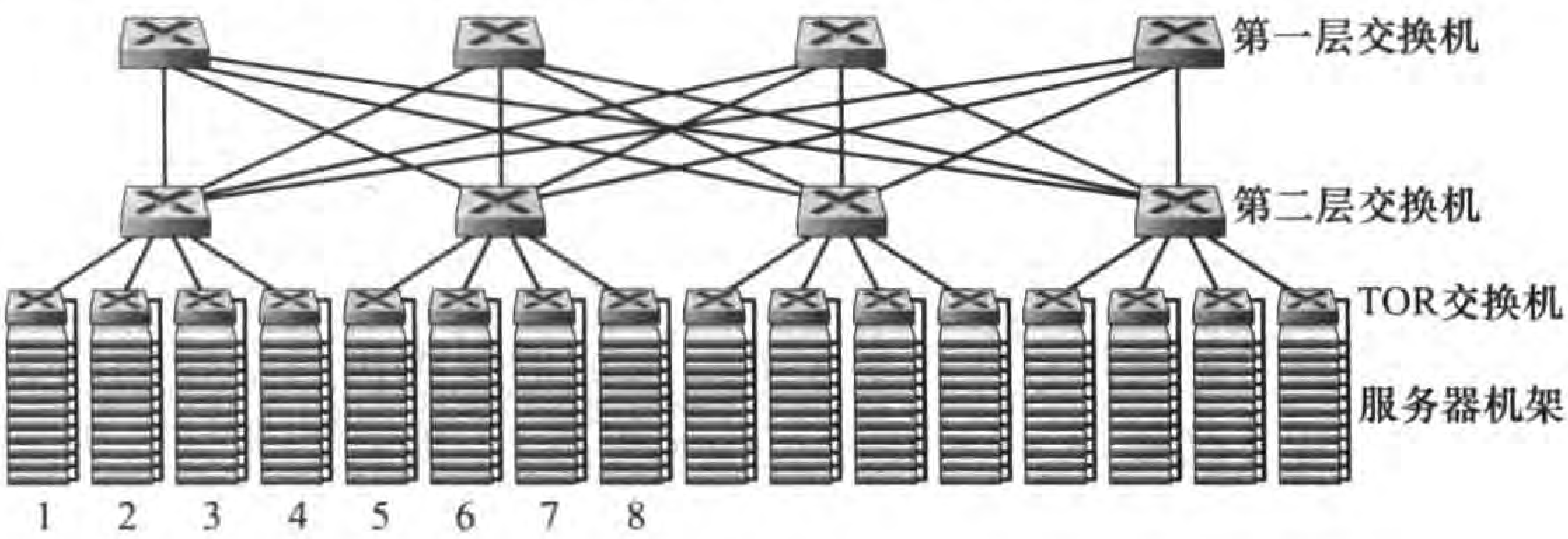


图6-31 高度互联的数据网络拓扑

另外一个主要的趋势就是采用基于船运集装箱的**模块化数据中心**（Modular Data Center，MDC）[You Tube 2009；Waldrop 2007]。在一个MDC中，在一个标准的12米船运集装箱内，工厂构建一个"迷你数据中心"并将该集装箱运送到数据中心的位置。每一个集装箱都有多达数千台主机，堆放在数十台机架上，并且紧密地排列在一起。在数据中心位置，多个集装箱彼此互联，同时也和因特网连接。一旦预制的集装箱部署在数据中心，通常难以检修。因此，每一个集装箱都得体地设计为性能下降：当组件（服务器和交换机）随着时间的推移出现故障时，集装箱继续运行但是性能下降。当许多组件出现故障并且性能已经下降到低于某个阈值时，整个集装箱将会被移除，并用新的来替换。

创建由集装箱构成的数据中心提出了新的联网挑战。对于MDC，有两种类型的网络：每一个集装箱中的内部网络和互联每个集装箱的核心网络［Guo 2009；Farrington 2010］。在每个集装箱内部，在规模上升到数千台主机的时候，通过廉价的商用吉比特以太网交换机创建全连接的网络（如前面所描述）是可行的。然而，核心网络的设计仍然是一个带有挑战性的问题，这需要能互联成百上千的集装箱，同时能够为典型工作负载提供跨多个集装箱的主机到主机间的高带宽。［Farrington 2010］中提出了一种互联集装箱的混合电/光交换机体系结构。

当采用高度互联拓扑的时候，一个主要的问题是设计交换机之间的路由选择算法。一种可能是采用随机路由选择方式［Greenberg 2009b］。另一种可能是在每台主机中部署多块网络接口卡［Guo 2009］，将每台主机连接到多台低成本的商用交换机上，并且允许主机自己在交换机间智能地为流量选路。这些方案的变种和扩展正被部署在当前的数据中心中。

另一种重要趋势是，大型云提供商正在其数据中心越来越多地建造或定制几乎所有东西，包括网络适配器、交换机路由器、TOR、软件和网络协议［Greenberg 2015；Singh 2015］。由亚马逊开创的另一个趋势是，用"可用性区域"来改善可靠性，这种技术在不同的邻近建筑物中基本上复制不同的数据中心。通过让建筑物邻近（几千米远），互相交互的数据能够跨越位于相同可用性区域的数据中心进行同步，与此同时提供容错性［Amazon 2014］。数据中心设计会不断出现更多的创新，感兴趣的读者可以查看近期的论文和有关数据中心设计的视频。

6.7 回顾：Web页面请求的历程

既然我们已经在本章中学过了链路层，并且在前面几章中学过了网络层、运输层和应用层，那么我们沿协议栈向下的旅程就完成了！在本书的一开始（1.1节），我们说过“本书的大部分内容与计算机网络协议有关”，在本章中，我们无疑已经看到了情况的确如此！在继续学习本书第二部分中时下关注的章节之前，通过对已经学过的协议做一个综合的、全面的展望，我们希望总结一下沿协议栈向下的旅程。而做这个“全面的”展望的一种方法是识别许多（许多!）协议，这些协议涉及满足甚至最简单的请求：下载一个Web页面。图6-32图示了我们的场景：一名学生Bob将他的便携机与学校的以太网交换机相连，下载一个Web页面（比如说www.google.com主页）。如我们所知，为满足这个看起来简单的请求，背后隐藏了许多细节。本章后面的Wireshark实验仔细检查了包含一些分组的踪迹文件，这些分组更为详细地涉及类似的场景。

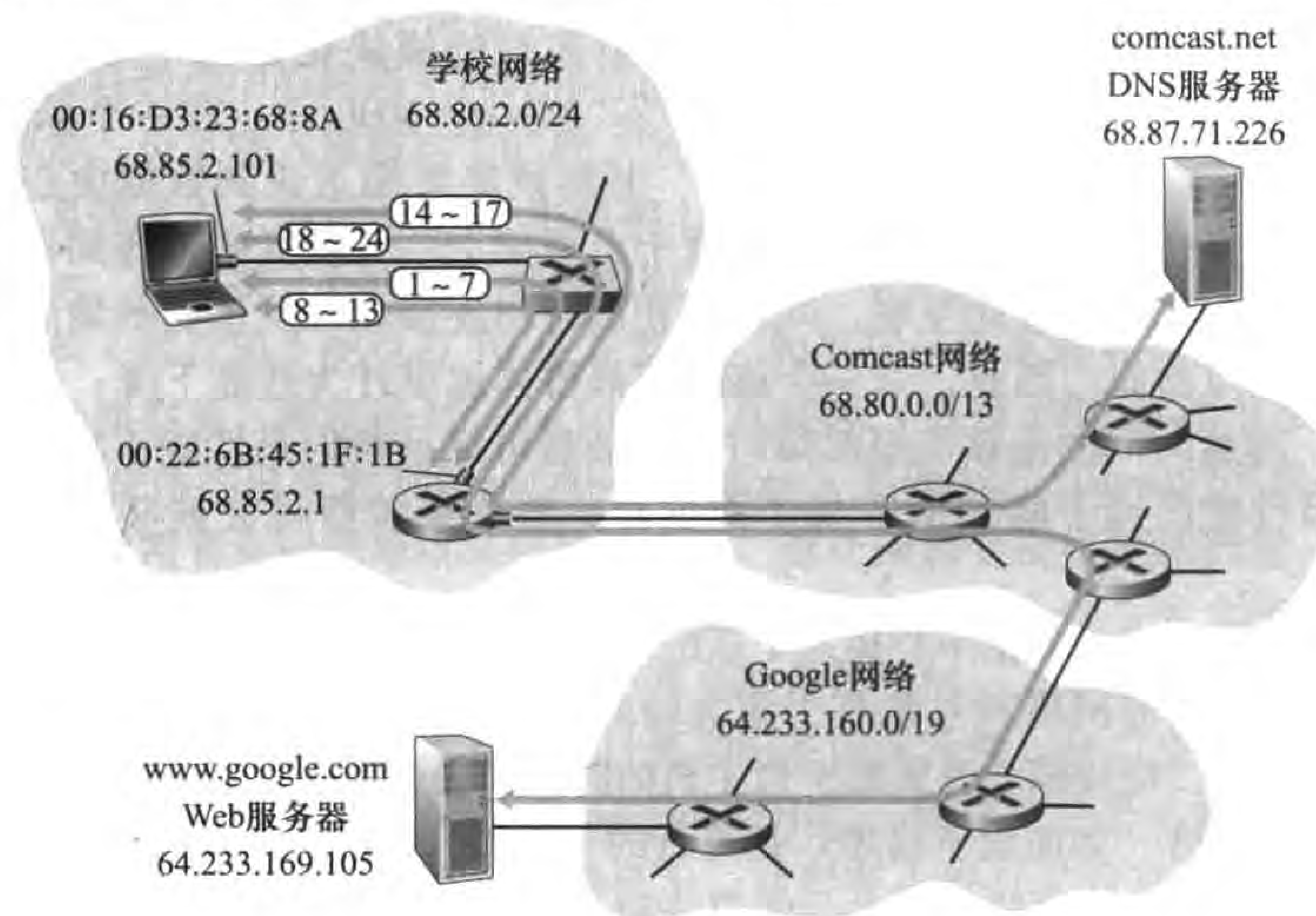


图6-32　Web页请求的历程：网络环境和动作

6.7.1 准备：DHCP、UDP、IP和以太网

我们假定Bob启动他的便携机，然后将其用一根以太网电缆连接到学校的以太网交换机，交换机又与学校的路由器相连，如图6-32所示。学校的这台路由器与一个ISP连接，本例中ISP为comcast.net。在本例中，comcast.net为学校提供了DNS服务；所以，DNS服务器驻留在Comcast网络中而不是学校网络中。我们将假设DHCP服务器运行在路由器中，就像常见情况那样。

当Bob首先将其便携机与网络连接时，没有IP地址他就不能做任何事情（例如下载一个Web网页）。所以，Bob的便携机所采取的一个网络相关的动作是运行DHCP协议，以从本地DHCP服务器获得一个IP地址以及其他信息。

1）Bob便携机上的操作系统生成一个**DHCP请求报文**（4.3.3节），并将这个报文放入具有目的端口67（DHCP服务器）和源端口68（DHCP客户）的**UDP报文段**（3.3节）

该 UDP 报文段则被放置在一个具有广播 IP 目的地址（255.255.255.255）和源 IP 地址 0.0.0.0 的 **IP 数据报**中（4.3.1 节），因为 Bob 的便携机还没有一个 IP 地址。

2）包含 DHCP 请求报文的 IP 数据报则被放置在**以太网帧**中（6.4.2 节）。该以太网帧具有目的 MAC 地址 FF:FF:FF:FF:FF:FF，使该帧将广播到与交换机连接的所有设备（如果顺利的话也包括 DHCP 服务器）；该帧的源 MAC 地址是 Bob 便携机的 MAC 地址 00:16:D3:23:68:8A。

3）包含 DHCP 请求的广播以太网帧是第一个由 Bob 便携机发送到以太网交换机的帧。该交换机在所有的出端口广播入帧，包括连接到路由器的端口。

4）路由器在它的具有 MAC 地址 00:22:6B:45:1F 的接口接收到该广播以太网帧，该帧中包含 DHCP 请求，并且从该以太网帧中抽取出 IP 数据报。该数据报的广播 IP 目的地址指示了这个 IP 数据报应当由在该节点的高层协议处理，因此该数据报的载荷（一个 UDP 报文段）**被分解**（3.2 节）向上到达 UDP，DHCP 请求报文从此 UDP 报文段中抽取出来。此时 DHCP 服务器有了 DHCP 请求报文。

5）我们假设运行在路由器中的 DHCP 服务器能够以 **CIDR**（4.3.3 节）块 68.85.2.0/24 分配 IP 地址。所以本例中，在学校内使用的所有 IP 地址都在 Comcast 的地址块中。我们假设 DHCP 服务器分配地址 68.85.2.101 给 Bob 的便携机。DHCP 服务器生成包含这个 IP 地址以及 DNS 服务器的 IP 地址（68.87.71.226）、默认网关路由器的 IP 地址（68.85.2.1）和子网块（68.85.2.0/24）（等价为“网络掩码”）的一个 DHCP ACK **报文**（4.3.3 节）。该 DHCP 报文被放入一个 UDP 报文段中，UDP 报文段被放入一个 IP 数据报中，IP 数据报再被放入一个以太网帧中。这个以太网帧的源 MAC 地址是路由器连到归属网络时接口的 MAC 地址（00:22:6B:45:1F:1B），目的 MAC 地址是 Bob 便携机的 MAC 地址（00:16:D3:23:68:8A）。

6）包含 DHCP ACK 的以太网帧由路由器发送给交换机。因为交换机是**自学习**的（6.4.3 节），并且先前从 Bob 便携机收到（包含 DHCP 请求的）以太网帧，所以该交换机知道寻址到 00:16:D3:23:68:8A 的帧仅从通向 Bob 便携机的输出端口转发。

7）Bob 便携机接收到包含 DHCP ACK 的以太网帧，从该以太网帧中抽取 IP 数据报，从 IP 数据报中抽取 UDP 报文段，从 UDP 报文段抽取 DHCP ACK 报文。Bob 的 DHCP 客户则记录下它的 IP 地址和它的 DNS 服务器的 IP 地址。它还在其 **IP 转发表**中安装默认网关的地址（4.1 节）。Bob 便携机将向该默认网关发送目的地址为其子网 68.85.2.0/24 以外的所有数据报。此时，Bob 便携机已经初始化好它的网络组件，并准备开始处理 Web 网页获取。（注意到在第 4 章给出的四个步骤中仅有最后两个 DHCP 步骤是实际必要的。）

6.7.2 仍在准备：DNS 和 ARP

当 Bob 将 www.google.com 的 URL 键入其 Web 浏览器时，他开启了一长串事件，这将导致谷歌主页最终显示在其 Web 浏览器上。Bob 的 Web 浏览器通过生成一个 **TCP 套接字**（2.7 节）开始了该过程，套接字用于向 www.google.com 发送 **HTTP 请求**（2.2 节）。为了生成该套接字，Bob 便携机将需要知道 www.google.com 的 IP 地址。我们在 2.4 节中学过，使用 **DNS 协议**提供这种名字到 IP 地址的转换服务。

8）Bob 便携机上的操作系统因此生成一个 **DNS 查询报文**（2.4.3 节），将字符串 www.google.com 放入 DNS 报文的问题段中。该 DNS 报文则放置在一个具有 53 号（DNS 服务器）目的端口的 UDP 报文段中。该 UDP 报文段则被放入具有 IP 目的地址 68.87.71.226（在

第5步中DHCP ACK返回的DNS服务器地址）和源IP地址68.85.2.101的IP数据报中。

9）Bob便携机则将包含DNS请求报文的数据报放入一个以太网帧中。该帧将发送（在链路层寻址）到Bob学校网络中的网关路由器。然而，即使Bob便携机经过上述第5步中的DHCP ACK报文知道了学校网关路由器的IP地址（68.85.2.1），但仍不知道该网关路由器的MAC地址。为了获得该网关路由器的MAC地址，Bob便携机将需要使用**ARP协议**（6.4.1节）。

10）Bob便携机生成一个具有目的IP地址68.85.2.1（默认网关）的**ARP查询报文**，将该ARP报文放置在一个具有广播目的地址（FF:FF:FF:FF:FF:FF）的以太网帧中，并向交换机发送该以太网帧，交换机将该帧交付给所有连接的设备，包括网关路由器。

11）网关路由器在通往学校网络的接口上接收到包含该ARP查询报文的帧，发现在ARP报文中目标IP地址68.85.2.1匹配其接口的IP地址。网关路由器因此准备一个**ARP回答**，指示它的MAC地址00:22:6B:45:1F:1B对应IP地址68.85.2.1。它将ARP回答放在一个以太网帧中，其目的地址为00:16:D3:23:68:8A（Bob便携机），并向交换机发送该帧，再由交换机将帧交付给Bob便携机。

12）Bob便携机接收包含ARP回答报文的帧，并从ARP回答报文中抽取网关路由器的MAC地址（00:22:6B:45:1F:1B）。

13）Bob便携机现在（最终！）能够使包含DNS查询的以太网帧寻址到网关路由器的MAC地址。注意到在该帧中的IP数据报具有IP目的地址68.87.71.226（DNS服务器），而该帧具有目的地址00:22:6B:45:1F:1B（网关路由器）。Bob便携机向交换机发送该帧，交换机将该帧交付给网关路由器。

6.7.3　仍在准备：域内路由选择到DNS服务器

14）网关路由器接收该帧并抽取包含DNS查询的IP数据报。路由器查找该数据报的目的地址（68.87.71.226），并根据其转发表决定该数据报应当发送到图6-32的Comcast网络中最左边的路由器。IP数据报放置在链路层帧中，该链路适合将学校路由器连接到最左边Comcast路由器，并且该帧经这条链路发送。

15）在Comcast网络中最左边的路由器接收到该帧，抽取IP数据报，检查该数据报的目的地址（68.87.71.226），并根据其转发表确定出接口，经过该接口朝着DNS服务器转发数据报，而转发表已根据Comcast的域内协议（如RIP、OSPF或IS-IS，5.3节）以及**因特网的域间协议BGP**（5.4节）所填写。

16）最终包含DNS查询的IP数据报到达了DNS服务器。DNS服务器抽取出DNS查询报文，在它的DNS数据库中查找名字www.google.com（2.4节），找到包含对应www.google.com的IP地址（64.233.169.105）的**DNS源记录**。（假设它当前缓存在DNS服务器中。）前面讲过这种缓存数据源于google.com的**权威DNS服务器**（2.4.2节）。该DNS服务器形成了一个包含这种主机名到IP地址映射的**DNS回答报文**，将该DNS回答报文放入UDP报文段中，该报文段放入寻址到Bob便携机（68.85.2.101）的IP数据报中。该数据报将通过Comcast网络反向转发到学校的路由器，并从这里经过以太网交换机到Bob便携机。

17）Bob便携机从DNS报文抽取出服务器www.google.com的IP地址。最终，在大量工作后，Bob便携机此时准备接触www.google.com服务器！

6.7.4 Web 客户-服务器交互：TCP 和 HTTP

18）既然 Bob 便携机有了 www.google.com 的 IP 地址，它能够生成 **TCP 套接字**（2.7 节），该套接字将用于向 www.google.com 发送 **HTTP GET** 报文（2.2.3 节）。当 Bob 生成 TCP 套接字时，在 Bob 便携机中的 TCP 必须首先与 www.google.com 中的 TCP 执行**三次握手**（3.5.6 节）。Bob 便携机因此首先生成一个具有目的端口 80（针对 HTTP 的）的 **TCP SYN** 报文段，将该 TCP 报文段放置在具有目的 IP 地址 64.233.169.105（www.google.com）的 IP 数据报中，将该数据报放置在 MAC 地址为 00:22:6B:45:1F:1B（网关路由器）的帧中，并向交换机发送该帧。

19）在学校网络、Comcast 网络和谷歌网络中的路由器朝着 www.google.com 转发包含 TCP SYN 的数据报，使用每台路由器中的转发表，如前面步骤 14～16 那样。前面讲过支配分组经 Comcast 和谷歌网络之间域间链路转发的路由器转发表项，是由 **BGP** 协议决定的（第 5 章）。

20）最终，包含 TCP SYN 的数据报到达 www.googole.com。从数据报抽取出 TCP SYN 报文并分解到与端口 80 相联系的欢迎套接字。对于谷歌 HTTP 服务器和 Bob 便携机之间的 TCP 连接生成一个连接套接字（2.7 节）。产生一个 TCP SYNACK（3.5.6 节）报文段，将其放入向 Bob 便携机寻址的一个数据报中，最后放入链路层帧中，该链路适合将 www.google.com 连接到其第一跳路由器。

21）包含 TCP SYNACK 报文段的数据报通过谷歌、Comcast 和学校网络，最终到达 Bob 便携机的以太网卡。数据报在操作系统中分解到步骤 18 生成的 TCP 套接字，从而进入连接状态。

22）借助于 Bob 便携机上的套接字，现在（终于！）准备向 www.google.com 发送字节了，Bob 的浏览器生成包含要获取的 URL 的 HTTP GET 报文（2.2.3 节）。HTTP GET 报文则写入套接字，其中 GET 报文成为一个 TCP 报文段的载荷。该 TCP 报文段放置进一个数据报中，并交付到 www.google.com，如前面步骤 18～20 所述。

23）在 www.google.com 的 HTTP 服务器从 TCP 套接字读取 HTTP GET 报文，生成一个 **HTTP 响应**报文（2.2 节），将请求的 Web 页内容放入 HTTP 响应体中，并将报文发送进 TCP 套接字中。

24）包含 HTTP 回答报文的数据报通过谷歌、Comcast 和学校网络转发，到达 Bob 便携机。Bob 的 Web 浏览器程序从套接字读取 HTTP 响应，从 HTTP 响应体中抽取 Web 网页的 html，并最终（终于！）显示了 Web 网页。

上面的场景已经涉及许多网络基础！如果你已经理解上面例子中的大多数或全部，则你也已经涵盖了许多基础知识，因为前面已经学过 1.1 节，其中我们谈道“本书的大部分内容与计算机网络协议有关”，并且你也许想知道一个协议实际是什么样子！上述例子看起来是尽可能详尽，我们已经忽略了一些可能的附加协议（例如，运行在学校网关路由器中的 NAT，到学校网络的无线接入，接入学校网络或对报文段或数据报加密的安全协议，网络管理协议），以及人们将会在公共因特网中遇到的一些考虑（Web 缓存，DNS 等级体系）。我们将在本书的第二部分涉及一些这类主题和更多内容。

最后，我们注意到上述例子是一个综合、完整的例子，还观察了本书第一部分所学习过的许多协议的十分“具体的细节”。该例子更多地关注“怎样做”而不是“为什么做”。

对于想开阔视野的读者来说，有关网络协议设计更为深思熟虑的一般观点可参见［Clark 1988；RFC 5218］。

6.8　小结

在这一章中，我们学习了链路层，包括它的服务、支撑它操作的原则和许多重要的特定协议，它们使用这些原则实现了链路层服务。

我们看到链路层的基本服务是将网络层的数据报从一个节点（主机、交换机、路由器，WiFi 接入点）移动到一个相邻的节点。我们看到，在通过链路向相邻节点传输之前，所有链路层协议都是通过将网络层数据报封装在链路层帧中来操作的。然而，除了这个共同的成帧功能之外，我们知道了不同的链路层协议提供截然不同的链路接入、交付和传输服务。造成这些差异的部分原因是链路层协议必须工作在很多种链路类型上。一个简单的点对点链路具有单个发送方和接收方，并通过单一“线路”通信。一个多路访问链路在许多发送方和接收方之间共享；因此，对多路访问信道的链路层协议有一个协调链路接入的协议（它的多路访问协议）。在 MPLS 的情况下，连接两个相邻节点（例如，在 IP 意义上的两台相邻的 IP 路由器，它们是到某个目的地的下一跳 IP 路由器）的“链路”，其本身可能实际上就是一个网络。从某种意义来说，将一个网络视为一条“链路”的想法没有什么可奇怪的。例如，连接家庭调制解调器/计算机到远端调制解调器/路由器的一条电话链路，实际上是一条穿过精密而复杂的电话网络的路径。

在链路层通信所依据的原理中，我们研究了差错检测和纠正技术、多路访问协议、链路层寻址、虚拟化（VLAN）以及扩展的交换局域网和数据中心网络的构造方法。今天对链路层的许多关注在于这些交换网络。在差错检测/纠正场景中，为了对帧通过链路传输时可能发生的比特翻转进行检测并在某些情况下进行纠正，我们研究了在帧的首部增加附加比特的方法。我们讨论了简单的奇偶校验和检验和方案，以及更健壮的循环冗余检测。然后我们转向多路访问协议主题。我们确定和学习了协调访问广播信道的 3 大类方法：信道划分方法（TDM、FDM）、随机接入方法（ALOHA 协议和 CSMA 协议）和轮流方法（轮询和令牌传递）。我们学习了电缆接入网，发现它使用了多种这些多路访问方法。我们看到让多个节点共享单个广播信道的结果，是需要在链路层提供节点地址。我们知道物理地址和网络层地址是非常不同的，而且在因特网场景中，一个专门的协议（ARP，即地址解析协议）用于在这两种寻址形式之间进行转换，并且详细学习了极为成功的以太网协议。然后我们研究了共享一个广播信道的节点是怎样形成一个局域网的，以及多个局域网怎样能够互联形成一个更大的局域网，即互联这些本地节点完全不需要网络层路由选择的干预。我们也知道了多个虚拟局域网是怎样能够产生一个单一的物理局域网体系结构的。

通过关注当 MPLS 网络互联 IP 路由器时是如何提供链路层服务的和展望今天用于大型数据中心的网络设计，我们结束了链路层的学习。通过识别在获取一个简单的 Web 网页时所需要的许多协议，我们完成了本章（和前 5 章）。在学习了链路层后，我们沿协议栈向下的旅程现在结束了！当然，物理层位于数据链路层之下，但是物理层的细节也许最好留给另外一门课程（例如，在通信理论而不是计算机网络课程中）去学习。然而我们在本章和第 1 章（在 1.2 节中讨论了物理媒体）中已经接触了物理层的几个方面。当我们在下一章中学习无线链路特性时，将再次考虑物理层。

尽管我们沿协议栈向下的旅程已结束，但我们计算机网络的学习仍然没有结束。在后面的 3 章中我们将讨论无线网络、网络安全和多媒体网络。这 3 个主题不便放进任何一层中；实际上每个主题跨越了多个层次。因此理解这些主题（在某些网络教材中被列为高级主题）需要对协议栈所有层次都有坚实的基础，我们对链路层的学习已经完成了这样的基础！

课后习题和问题

复习题

6.1 ~6.2 节

R1. 考虑在 6.1.1 节中的运输类比。如果一个乘客类比为一个数据报，什么类比于链路层帧？

R2. 如果在因特网中的所有链路都提供可靠的交付服务，TCP 可靠传输服务将是多余的吗？为什么？

R3. 链路层协议能够向网络层提供哪些可能的服务？在这些链路层服务中，哪些在 IP 中有对应的服务？哪些在 TCP 中有对应的服务？

6.3 节

R4. 假设两个节点同时经一个速率为 R 的广播信道开始传输一个长度为 L 的分组。用 d_{prop} 表示这两个节点之间的传播时延。如果 $d_{prop} < L/R$，会出现碰撞吗？为什么？

R5. 在 6.3 节中，我们列出了广播信道的 4 种希望的特性。这些特性中的哪些是时隙 ALOHA 所具有的？令牌传递具有这些特性中的哪些？

R6. 在 CSMA/CD 中，在第 5 次碰撞后，节点选择 $K=4$ 的概率有多大？结果 $K=4$ 在 10Mbps 以太网上对应于多少秒的时延？

R7. 使用人类在鸡尾酒会交互的类比来描述轮询和令牌传递协议。

R8. 如果局域网有很大的周长时，为什么令牌环协议将是低效的？

6.4 节

R9. MAC 地址空间有多大？IPv4 的地址空间呢？IPv6 的地址空间呢？

R10. 假设节点 A、B 和 C（通过它们的适配器）都连接到同一个广播局域网上。如果 A 向 B 发送数千个 IP 数据报，每个封装帧都有 B 的 MAC 地址，C 的适配器会处理这些帧吗？如果会，C 的适配器将会把这些帧中的 IP 数据报传递给 C 的网络层吗？如果 A 用 MAC 广播地址来发送这些帧，你的回答将有怎样的变化呢？

R11. ARP 查询为什么要在广播帧中发送呢？ARP 响应为什么要在一个具有特定目的 MAC 地址的帧中发送呢？

R12. 对于图 6-19 中的网络，路由器有两个 ARP 模块，每个都有自己的 ARP 表。同样的 MAC 地址可能在两张表中都出现吗？

R13. 比较 10BASE-T、100BASE-T 和吉比特以太网的帧结构。它们有什么不同吗？

R14. 考虑图 6-15。在 4.3 节的寻址意义下，有多少个子网呢？

R15. 在一个支持 802.1Q 协议交换机上能够配置的 VLAN 的最大数量是多少？为什么？

R16. 假设支持 K 个 VLAN 组的 N 台交换机经过一个干线协议连接起来。连接这些交换机需要多少端口？评价你的答案。

习题

P1. 假设某分组的信息内容是比特模式 1110 0110 1001 1101，并且使用了偶校验方案。在采用二维奇偶校验方案的情况下，包含该检验比特的字段的值是什么？你的回答应该使用最小长度检验和字段。

P2. 说明（举一个不同于图6-5的例子）二维奇偶校验能够纠正和检测单比特差错。说明（举一个例子）某些双比特差错能够被检测但不能纠正。

P3. 假设某分组的信息部分（图6-3中的D）包含10字节，它由字符串“Networking”的8比特无符号二进制ASCII表示组成。对该数据计算因特网检验和。

P4. 考虑前一个习题，但此时假设这10字节包含：

a. 数字1到10的二进制表示。

b. 字母B到K（大写）的ASCII表示。

c. 字母b到k（小写）的ASCII表示。

计算这些数据的因特网检验和。

P5. 考虑5比特生成多项式，$G=10011$，并且假设D的值为1010101010。R的值是什么？

P6. 考虑上一个习题，这时假设D具有值：

a. 1001010101。

b. 0101101010。

c. 1010100000。

P7. 在这道习题中，我们探讨CRC的某些性质。对于在6.2.3节中给出的生成多项式$G(=1001)$，回答下列问题：

a. 为什么它能够检测数据D中的任何单比特差错？

b. 上述G能够检测任何奇数比特差错吗？为什么？

P8. 在6.3节中，我们提供了时隙ALOHA效率推导的概要。在本习题中，我们将完成这个推导。

a. 前面讲过，当有N个活跃节点时，时隙ALOHA的效率是$Np(1-p)^{N-1}$。求出使这个表达式最大化的p值。

b. 使用在（a）中求出的p值，令N接近于无穷，求出时隙ALOHA的效率。（提示：当N接近于无穷时，$(1-1/N)^N$接近于$1/e$。）

P9. 说明纯ALOHA的最大效率是$1/(2e)$。注意：如果你完成了上面的习题，本习题就很简单了。

P10. 考虑两个节点A和B，它们都使用时隙ALOHA协议来竞争一个信道。假定节点A比节点B有更多的数据要传输，并且节点A的重传概率p_A比节点B的重传概率p_B要大。

a. 给出节点A的平均吞吐量的公式。具有这两个节点的协议的总体效率是多少？

b. 如果$p_A=2p_B$，节点A的平均吞吐量比节点B的要大两倍吗？为什么？如果不是，你能够选择什么样的p_A和p_B使得其成立？

c. 一般而言，假设有N个节点，其中的节点A具有重传概率$2p$并且所有其他节点具有重传概率p。给出表达式来计算节点A和其他任何节点的平均吞吐量。

P11. 假定4个活跃节点A、B、C和D都使用时隙ALOHA来竞争访问某信道。假设每个节点有无限个分组要发送。每个节点在每个时隙中以概率p尝试传输。第一个时隙编号为时隙1，第二个时隙编号为时隙2，等等。

a. 节点A在时隙5中首先成功的概率是多少？

b. 某个节点（A、B、C或D）在时隙4中成功的概率是多少？

c. 在时隙3中出现首个成功的概率是多少？

d. 这个4节点系统的效率是多少？

P12. 对N的下列值，画出以p为函数的时隙ALOHA和纯ALOHA的效率。

a. $N=15$。

b. $N=25$。

c. $N=35$。

P13. 考虑具有N个节点和传输速率为R bps的一个广播信道。假设该广播信道使用轮询进行多路访问（有一个附加的轮询节点）。假设从某节点完成传输到后续节点允许传输之间的时间量（即轮询时延）是d_{poll}。假设在一个轮询周期中，一个给定的节点允许至多传输Q比特。该广播信道的最大吞

吐量是多少?

P14. 如图6-33所示，考虑通过两台路由器互联的3个局域网。

a. 对所有的接口分配IP地址。对子网1使用形式为192.168.1.xxx的地址，对子网2使用形式为192.168.2.xxx的地址，对子网3使用形式为192.168.3.xxx的地址。

b. 为所有的适配器分配MAC地址。

c. 考虑从主机E向主机B发送一个IP数据报。假设所有的ARP表都是最新的。就像在6.4.1节中对单路由器例子所做的那样，列举出所有步骤。

d. 重复(c)，现在假设在发送主机中的ARP表为空(并且其他表都是最新的)。

P15. 考虑图6-33。现在我们用一台交换机代替子网1和子网2之间的路由器，并且将子网2和子网3之间的路由器标记为R1。

a. 考虑从主机E向主机F发送一个IP数据报。主机E将请求路由器R1帮助转发该数据报吗？为什么？在包含IP数据报的以太网帧中，源和目的IP和MAC地址分别是什么？

b. 假定E希望向B发送一个IP数据报，假设E的ARP缓存中不包含B的MAC地址。E将执行ARP查询来发现B的MAC地址吗？为什么？在交付给路由器R1的以太网帧(包含发向B的IP数据报)中，源和目的IP和MAC地址分别是什么？

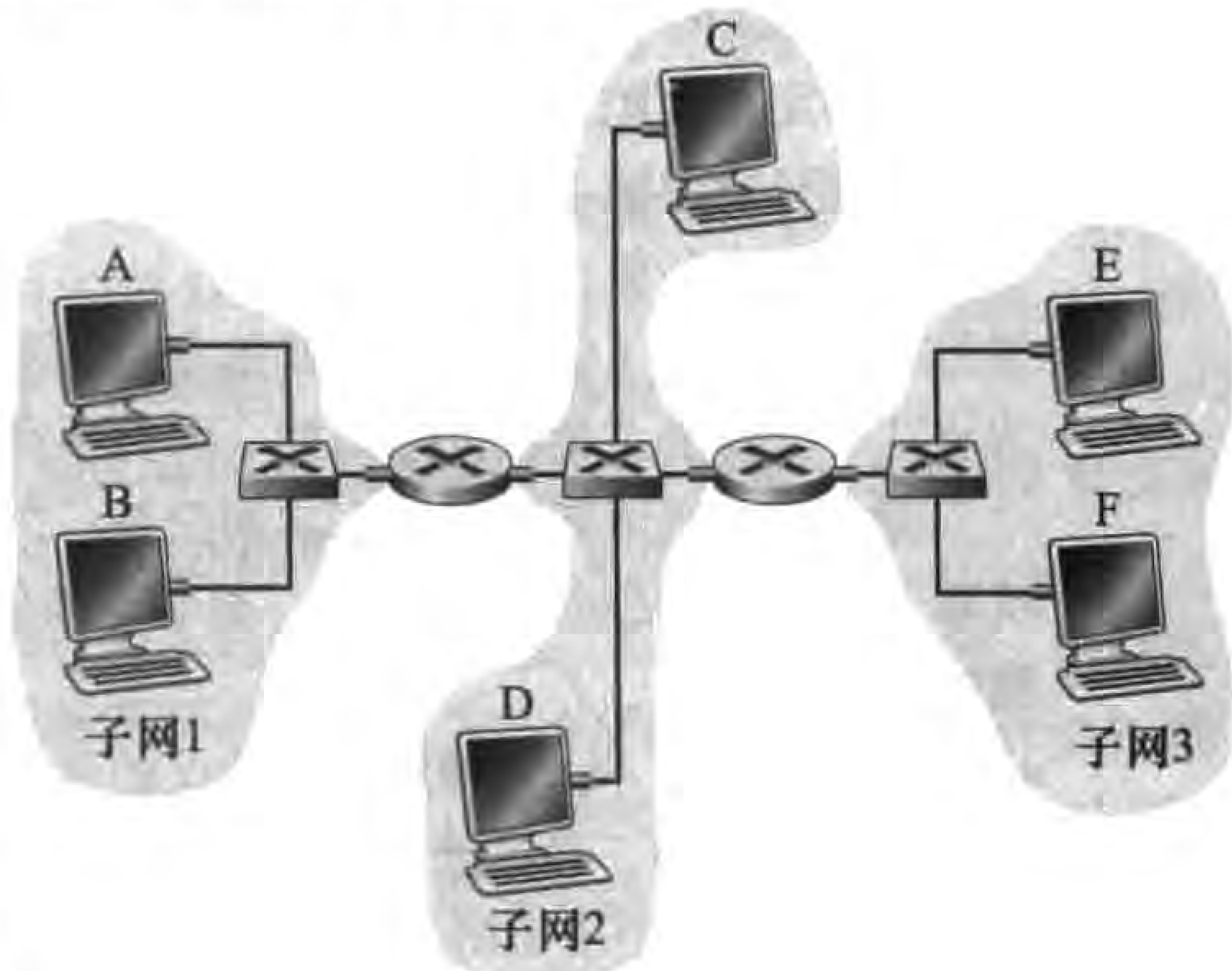


图6-33 由路由器互联的3个子网

c. 假定主机A希望向主机B发送一个IP数据报，A的ARP缓存不包含B的MAC地址，B的ARP缓存也不包含A的MAC地址。进一步假定交换机S1的转发表仅包含主机B和路由器R1的表项。因此，A将广播一个ARP请求报文。一旦交换机S1收到ARP请求报文将执行什么动作？路由器R1也会收到这个ARP请求报文吗？如果收到的话，R1将向子网3转发该报文吗？一旦主机B收到这个ARP请求报文，它将向主机A回发一个ARP响应报文。但是它将发送一个ARP查询报文来请求A的MAC地址吗？为什么？一旦交换机S1收到来自主机B的一个ARP响应报文，它将做什么？

P16. 考虑前面的习题，但是现在假设用一台交换机代替子网2和子网3之间的路由器，在这种新的场景中回答前面习题中的问题(a)～(c)。

P17. 前面讲过，使用CSMA/CD协议，适配器在碰撞之后等待 $K \cdot 512$ 比特时间，其中 K 是随机选取的。对于 $K=100$，对于一个10Mbps的广播信道，适配器返回到第二步要等多长时间？对于100Mbps的广播信道来说呢？

P18. 假设节点A和节点B在同一个10Mbps广播信道上，这两个节点的传播时延为325比特时间。假设对这个广播信道使用CSMA/CD和以太网分组。假设节点A开始传输一帧，并且在它传输结束之前节点B开始传输一帧。在A检测到B已经传输之前，A能完成传输吗？为什么？如果回答是可以，则A错误地认为它的帧已成功传输而无碰撞。提示：假设在 $t=0$ 比特时刻，A开始传输一帧。在最坏的情况下，A传输一个512+64比特时间的最小长度的帧。因此A将在 $t=512+64$ 比特时刻完成帧的传输。如果B的信号在比特时间 $t=512+64$ 比特之前到达A，则答案是否定的。在最坏的情况下，B的信号什么时候到达A?

P19. 假设节点A和节点B在相同的10Mbps广播信道上，并且这两个节点的传播时延为245比特时间。假设A和B同时发送以太网帧，帧发生了碰撞，然后A和B在CSMA/CD算法中选择不同的 K 值。假设没有其他节点处于活跃状态，来自A和B的重传会碰撞吗？为此，完成下面的例子就足以说明问题了。假设A和B在 $t=0$ 比特时间开始传输。它们在 $t=245$ 比特时间都检测到了碰撞。假设

$K_A=0$，$K_B=1$。B 会将它的重传调整到什么时间？A 在什么时间开始发送？（注意：这些节点在返回第 2 步之后，必须等待一个空闲信道，参见协议。）A 的信号在什么时间到达 B 呢？B 在它预定的时刻抑制传输吗？

P20. 在这个习题中，你将对一个类似于 CSMA/CD 的多路访问协议的效率进行推导。在这个协议中，时间分为时隙，并且所有适配器都与时隙同步。然而，和时隙 ALOHA 不同的是，一个时隙的长度（以秒计）比一帧的时间（即传输一帧的时间）小得多。令 S 表示一个时隙的长度。假设所有帧都有恒定长度 $L=kRS$，其中 R 是信道的传输速率，k 是一个大整数。假定有 N 个节点，每个节点都有无穷多帧要发送。我们还假设 $d_{prop}<S$，以便所有节点在一个时隙时间结束之前能够检测到碰撞。这个协议描述如下：

- 对于某给定的时隙，如果没有节点占有这个信道，所有节点竞争该信道；特别是每个节点以概率 p 在该时隙传输。如果刚好有一个节点在该时隙中传输，该节点在后续的 $k-1$ 个时隙占有信道，并传输它的整个帧。
- 如果某节点占用了信道，所有其他节点抑制传输，直到占有信道的这个节点完成了该帧的传输为止。一旦该节点传输完它的帧，所有节点竞争该信道。

注意到此信道在两种状态之间交替："生产性状态"（它恰好持续 k 个时隙）和"非生产性状态"（它持续随机数个时隙）。显然，该信道的效率是 $k/(k+x)$，其中 x 是连续的非生产性时隙的期望值。

a. 对于固定的 N 和 p，确定这个协议的效率。
b. 对于固定的 N，确定使该效率最大化的 p 值。
c. 使用在（b）中求出的 p（它是 N 的函数），确定当 N 趋向无穷时的效率。
d. 说明随着帧长度变大，该效率趋近于 1。

P21. 现在考虑习题 P14 中的图 6-33。对主机 A、两台路由器和主机 F 的各个接口提供 MAC 地址和 IP 地址。假定主机 A 向主机 F 发送一个数据报。当在下列场合传输该帧时，给出在封装该 IP 数据报的帧中的源和目的 MAC 地址：（i）从 A 到左边的路由器；（ii）从左边的路由器到右边的路由器；（iii）从右边的路由器到 F。还要给出到达每个点时封装在该帧中的 IP 数据报中的源和目的 IP 地址。

P22. 现在假定在图 6-33 最左边的路由器被一台交换机替换。主机 A、B、C 和 D 和右边的路由器以星形方式与这台交换机相连。当在下列场合传输该帧时，给出在封装该 IP 数据报的帧中的源和目的 MAC 地址：（i）从 A 到左边路由器；（ii）从左边路由器到右边的路由器；（iii）从右边的路由器到 F。还要给出到达每个点时封装在该帧中的 IP 数据报中源和目的 IP 地址。

P23. 考虑图 6-15。假定所有链路都是 100Mbps。在该网络中的 9 台主机和两台服务器之间，能够取得的最大总聚合吞吐量是多少？你能够假设任何主机或服务器能够向任何其他主机或服务器发送分组。为什么？

P24. 假定在图 6-15 中的 3 台连接各系的交换机用集线器来代替。所有链路是 100Mbps。现在回答习题 P23 中提出的问题。

P25. 假定在图 6-15 中的所有交换机用集线器来代替。所有链路是 100Mbps。现在回答在习题 P23 中提出的问题。

P26. 在某网络中标识为 A 到 F 的 6 个节点以星形与一台交换机连接，考虑在该网络环境中某个正在学习的交换机的运行情况。假定：（i）B 向 E 发送一个帧；（ii）E 向 B 回答一个帧；（iii）A 向 B 发送一个帧；（iv）B 向 A 回答一个帧。该交换机表初始为空。显示在这些事件的前后该交换机表的状态。对于每个事件，指出在其上面转发传输的帧的链路，并简要地评价你的答案。

P27. 在这个习题中，我们探讨用于 IP 语音应用的小分组。小分组长度的一个主要缺点是链路带宽的较大比例被首部字节所消耗。基于此，假定分组是由 P 字节和 5 字节首部组成。

a. 考虑直接发送一个数字编码语音源。假定该源以 128kbps 的恒定速率进行编码。假设每个源向网

络发送分组之前每个分组被完全填充。填充一个分组所需的时间是**分组化时延**（packetization delay）。根据 L，确定分组化时延（以毫秒计）。

b. 大于 20 毫秒的分组化时延会导致一个明显的、令人不快的回音。对于 $L=1500$ 字节（大致对应于一个最大长度的以太网分组）和 $L=50$ 字节（对应于一个 ATM 信元），确定该分组化时延。

c. 对 $R=622$Mbps 的链路速率以及 $L=1500$ 字节和 $L=50$ 字节，计算单台交换机的存储转发时延。

d. 对使用小分组长度的优点进行评述。

P28. 考虑图 6-25 中的单个交换 VLAN，假定一台外部路由器与交换机端口 1 相连。为 EE 和 CS 的主机和路由器接口分配 IP 地址。跟踪从 EE 主机向 CS 主机传送一个数据报时网络层和链路层所采取的步骤（提示：重读课文中对图 6-19 的讨论）。

P29. 考虑显示在图 6-29 中的 MPLS 网络，假定路由器 R5 和 R6 现在是 MPLS 使能的。假定我们要执行流量工程，使从 R6 发往 A 的分组要经 R6-R4-R3-R1 交换到 A，从 R5 发向 A 的分组要过 R5-R4-R2-R1 交换。给出 R5 和 R6 中的 MPLS 表以及在 R4 中修改的表，使得这些成为可能。

P30. 再次考虑上一个习题中相同的场景，但假定从 R6 发往 D 的分组经 R6-R4-R3 交换，而从 R5 发往 D 的分组经 R4-R2-R1-R3 交换。说明为使这些成为可能在所有路由器中的 MPLS 表。

P31. 在这个习题中，你将把已经学习过的因特网协议的许多东西拼装在一起。假设你走进房间，与以太网连接，并下载一个 Web 页面。从打开 PC 电源到得到 Web 网页，发生的所有协议步骤是什么？假设当你给 PC 加电时，在 DNS 或浏览器缓存中什么也没有。（提示：步骤包括使用以太网、DHCP、ARP、DNS、TCP 和 HTTP 协议。）明确指出在这些步骤中你如何获得网关路由器的 IP 和 MAC 地址。

P32. 考虑在图 6-30 中具有等级拓扑的数据中心网络。假设现在有 80 对流，在第 1 和第 9 机架之间有 10 个流，在第 2 和第 10 机架之间有 10 个流，等等。进一步假设网络中的所有链路是 10Gbps，而主机和 TOR 交换机之间的链路是 1Gbps。

a. 每条流具有相同的数据率；确定一条流的最大速率。

b. 对于相同的流量模式，对于图 6-31 中高度互联的拓扑，确定一条流的最大速率。

c. 现在假设有类似的流量模式，但在每个机架上涉及 20 台主机和 160 对流。确定对这两个拓扑的最大流速率。

P33. 考虑图 6-30 中所示的等级网络，并假设该数据中心需要在其他应用程序之间支持电子邮件和视频分发。假定 4 个服务器机架预留用于电子邮件，4 个服务器机架预留用于视频。对于每个应用，所有 4 个机架必须位于某单一的二层交换机之下，因为二层到一层链路没有充足的带宽来支持应用内部的流量。对于电子邮件应用，假定 99.9% 时间仅使用 3 个机架，并且视频应用具有相同的使用模式。

a. 电子邮件应用需要使用第 4 个机架的时间比例有多大？视频应用需要使用第 4 个机架的时间比例有多大？

b. 假设电子邮件使用和视频使用是独立的，这两个应用需要其第 4 个机架的时间比例有多大（等价地，概率有多大）？

c. 假设对于一个应用服务器短缺的时间具为 0.001% 或更少（引起用户在极短时间内性能恶化）。讨论在图 6-31 中的拓扑能够怎样使用，使得仅 7 个机架被共同地分配给两个应用（假设拓扑能够支持所有流量）。

Wireshark 实验

在与本教科书配套的 Web 站点（http://www.pearsonhighered.com/cs-resources/）上，你将找到一个 Wireshark 实验，该实验研究了 IEEE 802.3 协议的操作和以太网帧格式。第二个 Wireshark 实验研究了在家庭网络场景下所获取的分组踪迹。

人物专访

Simon S. Lam 是位于奥斯丁的T得克萨斯大学计算机科学系的教授和董事会主席。1971～1974年，他在UCLA的ARPA网络测量中心工作，从事卫星和无线电分组交换方面的工作。他领导的研究组于1993年发明了安全套接字及其原型，这是第一个安全套接字层（称为安全网络编程），为此赢得了2004年ACM软件系统奖。他的研究兴趣在于网络协议和安全服务的设计和分析。他从华盛顿州立大学获得了电气工程学士（BSEE）学位，从UCLA获得了硕士和博士学位。他于2007年当选美国国家工程院院士。



Simon S. Lam

- 为什么您决定专注于网络？

当我于1969年秋季作为一名研究生新生来到UCLA时，我原来的意图是研究控制论。后来我参加了Leonard Kleinrock的排队论课程，他给我留下了深刻印象。不久后，我以排队系统的自适应控制作为可能的论文题目进行了研究。在1972年初，Larry Roberts启动了ARPAnet卫星系统项目（后来称为分组卫星）。Kleinrock教授请我参加该项目。我们做的第一件事是为时隙ALOHA协议引入一个简单而实用的回退算法。不久后，我发现了许多有趣的研究问题，如ALOHA的不稳定性问题和对自适应回退的需求，这些形成了我学位论文的核心。

- 20世纪70年代您开始了在UCLA的学生生涯，活跃于因特网的早期阶段。那时的情况怎样？那时人们对因特网将变成什么样有哪些大致想法？

那时的环境与我曾在产业界和学术界所见到的其他系统构建项目的确没什么不同。ARPAnet初始规定的目标是相当谨慎的，即提供从远程位置接入昂贵的计算机的手段，使许多科学家能够使用这些计算机。然而，随着分组卫星项目于1972年和分组无线电项目于1973年启动，ARPA的目标有了相当大的扩展。到了1973年，ARPA同时建造了3个不同的分组网络，对Vint Cerf和Bob Kahn来说研发互联策略变得有必要了。

回顾过去，所有这些网络方面的进步和进展被视为（我相信）符合逻辑的而不是虚幻的。没有人能够想象到今天的因特网规模和个人计算机的能力。在第一台PC出现之前有十多年时间。那时为了正确地做处理，多数学生以一系列穿孔卡片的形式提交他们的计算机程序以进行批处理。仅有某些学生能够直接接触计算机，而计算机通常位于一个受限的区域中。调制解调器速度缓慢且稀有。作为一名研究生，在书桌上我仅有一部电话机，使用铅笔和纸从事我的大部分工作。

- 您认为网络和因特网未来将往何处发展？

在过去，因特网IP协议的简单性是它在竞争上取胜的最大法宝，并且成为网络互联事实上的标准。与它的竞争对手如20世纪80年代的X.25和20世纪90年代的ATM不同，IP能够在任何链路层联网技术之上运行，因为它仅提供尽力而为的数据报服务。因此，任何分组网络都能够连接到因特网上。

今天，IP的最大长处实际上成了一种缺点。IP目前像一件紧身衣，限制了因特网向特定方向发展。近年来，许多研究人员仅将他们的努力重新定向到应用层上。在无线自组织网络、传感网络和卫星网络方面也做了大量研究工作。这些网络能够看作独立的系统或链路层系统，因为位于IP的紧身衣之外，所以它们能够繁荣发展。

许多人对于P2P系统可能作为新奇的因特网应用的平台而感到欢欣鼓舞。然而，P2P系统在使用因特网资源方面效率很低。我关注的一个问题是，随着因特网不断互联各种设备和支持未来的P2P使能的应用，因特网核心的传输和交换能力比起对因特网流量的需求来说，前者是否将更快地不断增加。没有大量的容量预留，在面临恶意攻击和拥塞时，确保网络稳定性将是一项重要的任务。

因特网的急剧增长也要求以高速率为全世界的网络操作者和企业分配新的IP地址。以当前速度，未分配的IP地址池将在几年内消耗殆尽。当该事件发生时，仅能从IPv6地址空间中分配大量地址空间

的相邻块了。因为IPv6采用的启动缓慢，由于对早期用户缺乏激励，所以IPv4和IPv6将很有可能在今后的因特网上共存很多年。从IPv4主宰的因特网向IPv6主宰的因特网迁移将需要全球做出巨大的努力。

- 您的工作最具挑战性的部分是什么?

我的工作最具挑战性的部分是，作为一名教授传授和激发我课堂上的每个学生以及我所指导的每个博士研究生，而不只是获得重要的成就。非常聪明和有学习动力的人可能要求少量的引导而非许多其他东西。我经常从学生那里学到很多东西，比他们从我这里学到的多。教育和激励学术上落后的学生是一项重要的挑战。

- 您能预见到技术对未来学习方面有哪些影响?

最终几乎所有的人类知识将可以通过因特网得到，因特网将成为最为强有力的学习工具。这种巨大的知识库将具有为全世界的学生们提供公平竞争环境的潜力。例如，任何国家的有上进心的学生将能够访问最好课程的Web站点、多媒体演讲和教学材料。据说IEEE和ACM数字图书馆已经加速了中国的计算机科学研究的发展。与此同时，因特网将超越所有学习的地理障碍。

第7章

Computer Networking: A Top-Down Approach, Seventh Edition

无线网络和移动网络

在电话技术领域，过去的20年无疑可认为是蜂窝电话技术的黄金发展期。全球范围的移动蜂窝电话用户数量从1993年的3400万增长到2014年的超过70亿，同时蜂窝电话用户数量现在也超过了有线电话用户数量。现在移动电话的订购数量比我们地球上的人口数量还要多。蜂窝电话的许多优点是显而易见的，通过一个移动性强、重量轻的设备，能在任何地方、任何时间无缝地接入全球电话网络。最近，便携机、智能手机和平板电脑能够以无线方式经过蜂窝网或WiFi网络连接到因特网，而且，如游戏机、恒温调节器、家庭安全系统、家用器具、手表、眼镜、汽车、流量控制系统等设备也越来越多地以无线方式连接到因特网。

从网络的观点来说，由这些无线和移动设备联网所引发的挑战，特别是在数据链路层和网络层，与传统的有线网络的差别非常大，适合用单独一章（即本章）的篇幅来专门讨论无线网络和移动网络。

在本章中，我们首先讨论移动用户、无线链路和网络，以及它们与所连接的更大网络（通常是有线网络）之间的关系。我们将指出以下两方面的差别：一个是在该网络中由通信链路的*无线*特性所带来的挑战，另一个是由这些无线链路使能的*移动性*。在无线和移动性之间做重要区分，使我们能更好地区分、标识和掌握在每个领域中的重要概念。值得注意的是，确实在许多网络环境中网络节点是无线的而不是移动的（例如具有固定工作站和大显示器的无线家庭或办公网络），而有限的移动性也并不要求用无线链路（例如，一个在家里使用便携机的员工，关闭便携机，开车去工作，然后将该便携机连接到公司的有线网络上）。当然，许多最让人激动的网络环境是那些让用户同时具有无线和移动性的网络环境。例如在下列情况下，一个移动用户（不妨假设其正坐在汽车后座上）维持一个IP语音呼叫和多个进行中的TCP连接，同时又以每小时160公里的速度飞驰在高速公路上。正是在这种具有无线和移动的交集的场合中，我们将会发现最有趣的技术挑战。

首先我们将说明用于考虑无线通信和移动性的环境，即无线（并可能是移动）用户通过位于网络边缘的无线链路连接进更大的网络基础设施的网络。然后在7.2节中，我们考虑这种无线链路的特性，还将包括一个对码分多址接入（Code Division Multiple Access，CDMA）的简要介绍。CDMA是一个在无线网络中经常使用的共享媒体接入协议。在7.3节中，我们将更为深入地分析IEEE 802.11（WiFi）无线LAN标准的链路级方面；同时我们还将对蓝牙和其他无线个人域网络做简要描述。在7.4节中，我们概述蜂窝因特网接入，其中包括3G和正在兴起的4G蜂窝技术，这些技术能够同时提供语音和高速因特网接入。在7.5节，我们将注意力转向移动性，关注于移动用户的定位问题、对移动用户的路由选择以及“切换”（handing off）移动用户（即在网络中从一个接入点动态地移动到另一点的用户）等问题。在7.6节和7.7节中，我们将分别考察如何在企业802.11网络和LTE蜂窝网络中用移动IP标准实现这些移动服务。最后，我们将在7.8节中考虑无线链路和移动性对运输层协议和网络应用程序的影响。

7.1 概述

图 7-1 显示了我们将要讨论无线数据通信和移动性主题的环境。为使讨论具有一般性以覆盖各种各样的网络，将包括像 IEEE 802.11 这样的无线局域网和像 4G 网络这样的蜂窝网络来开始我们的讨论；然后在后续各节中，我们将对特定的无线体系结构进行更加详细的讨论。我们能够看到无线网络的下列要素：

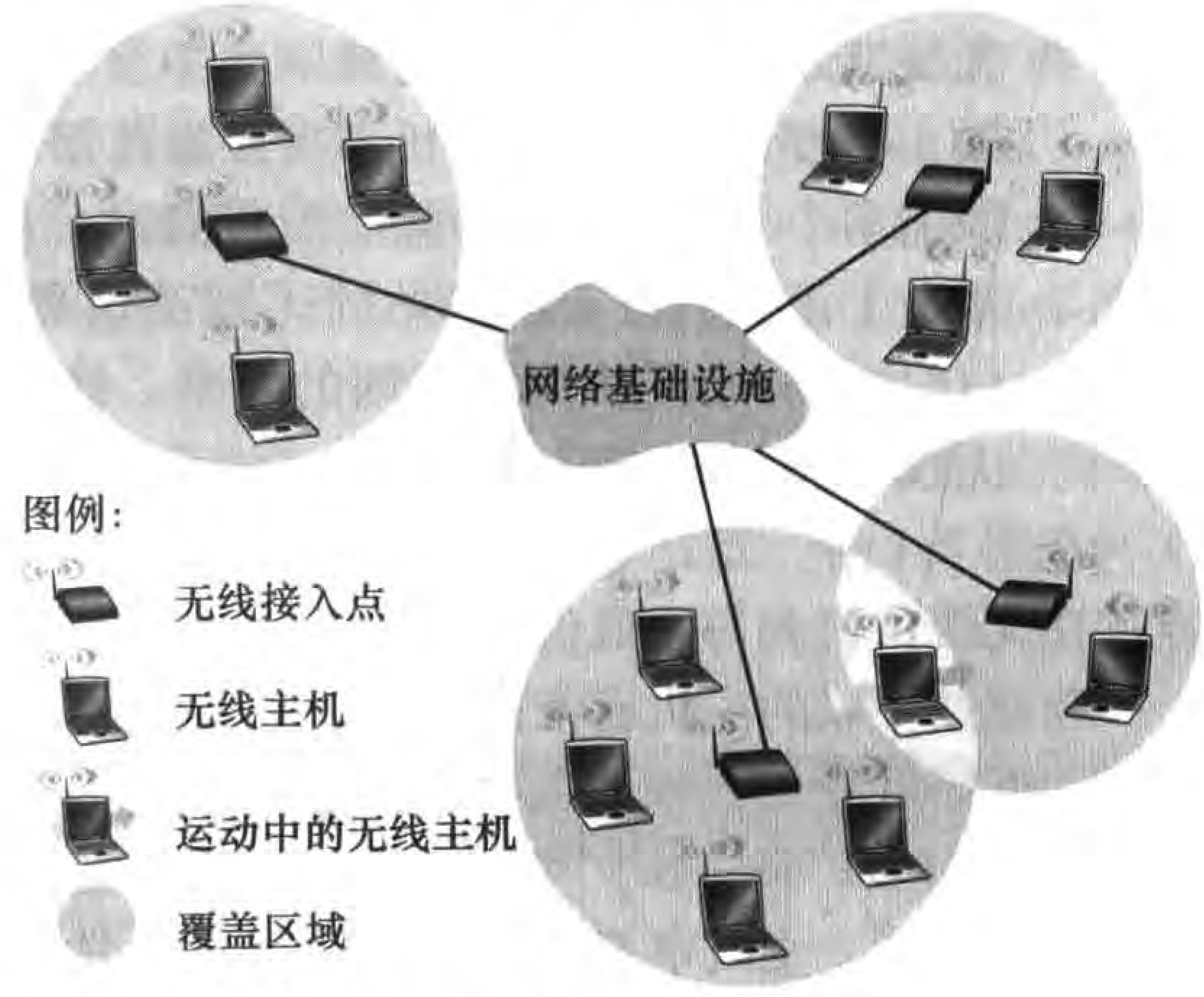


图 7-1 无线网络的要素

- 无线主机。如同在有线网络中一样，主机是运行应用程序的端系统设备。**无线主机**（wireless host）可以是便携机、掌上机、智能手机或者桌面计算机。主机本身可能移动，也可能不移动。
- 无线链路。主机通过**无线通信链路**（wireless communication link）连接到一个基站（定义见下文）或者另一台无线主机。不同的无线链路技术具有不同的传输速率和能够传输不同的距离。图 7-2 显示了较为流行的无线链路标准的两种主要特性（覆盖区域和链路速率）。（该图仅表示了提供这些特性的大致概念。例如，这些类型中的某些网络现在只是在部署，某些链路速率取决于距离、信道条件和在无线网络中的用户数量，能够比显示的值更高或更低些。）我们将在本章的前半部分讨论这些标准。在 7.2 节中，我们也考虑其他无线链路特性（如它们的比特差错率及其原因）。

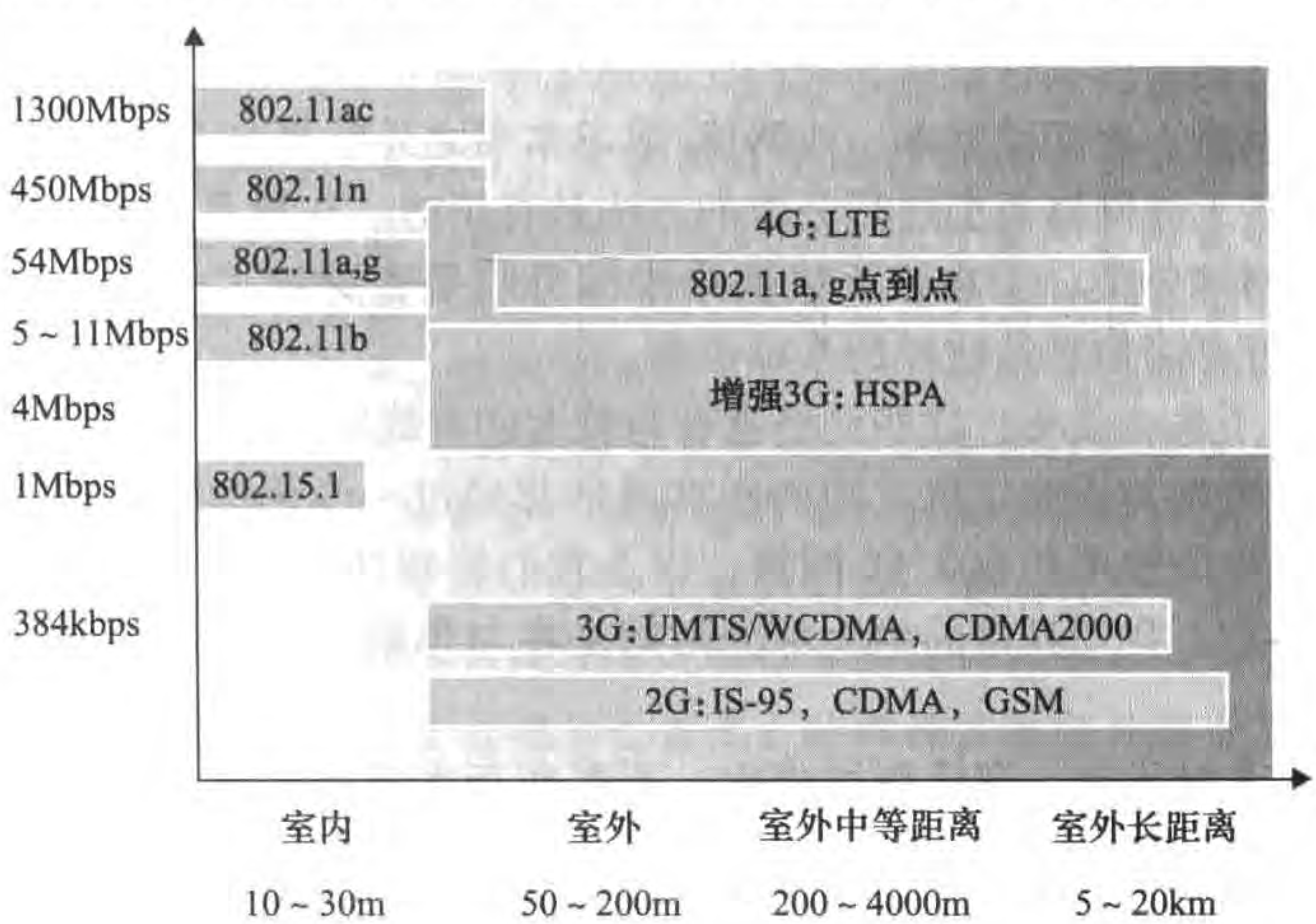


图 7-2 部分无线网络标准的链路特性

在图 7-1 中，无线链路将位于网络边缘的主机连接到更大的网络基础设施中。我

们要马上补充的是，无线链路有时同样应用在一个网络中以连接路由器、交换机和其他网络设备。然而，在本章中我们关注的焦点是无线通信在网络边缘的应用，因为许多最为振奋人心的技术挑战和多数增长就发生在这里。

- 基站。**基站**（base station）是无线网络基础设施的一个关键部分。与无线主机和无线链路不同，基站在有线网络中没有明确的对应设备。它负责向与之关联的无线主机发送数据和从主机那里接收数据（例如分组）。基站通常负责协调与之相关联的多个无线主机的传输。当我们说一台无线主机与某基站“相关联”时，则是指：①该主机位于该基站的无线通信覆盖范围内；②该主机使用该基站中继它（该主机）和更大网络之间的数据。蜂窝网络中的**蜂窝塔**（cell tower）和802.11无线LAN中的**接入点**（access point）都是基站的例子。

在图7-1中，基站与更大网络相连（如因特网、公司或家庭网、电话网），因此这种连接在无线主机和与之通信的其他部分之间起着链路层中继的作用。

与基站关联的主机通常被称为以**基础设施模式**（infrastructure mode）运行，因为所有传统的网络服务（如地址分配和路由选择）都由网络向通过基站相连的主机提供。在**自组织网络**（ad hoc network）中，无线主机没有这样的基础设施与之相连。在没有这样的基础设施的情况下，主机本身必须提供诸如路由选择、地址分配、类似于DNS的名字转换等服务。

当一台移动主机的移动超出一个基站的覆盖范围而到达另一个基站的覆盖范围后，它将改变其接入到更大网络的连接点（即改变与之相关联的基站），这一过程称作**切换**（handoff）。这种移动性引发了许多具有挑战性的问题。如果一台主机可以移动，那么如何找到它在网络中的当前位置，从而使得数据可以向该移动主机转发？如果一台主机可以位于许多可能位置中的一个，那么如何进行编址？如果主机在一个TCP连接或者电话呼叫期间移动，数据如何路由选择而使连接保持不中断？这些以及许多（许多！）其他问题使得无线网络和移动网络成为一个让人振奋的网络研究领域。

- 网络基础设施。这是无线主机希望与之进行通信的更大网络。

在讨论完无线网络的构件以后，我们注意到这些构件能够以多种不同方式组合以形成不同类型的无线网络。当阅读本章，或阅读/学习本书之外的更多有关无线网络的内容时，你可能发现对这些无线网络类型进行分类的方法是有用的。在最高层次，我们能够根据两个准则来对无线网络分类：①在该无线网络中的分组是否跨越了一个*无线跳或多个无线跳*；②网络中是否有诸如基站这样的*基础设施*。

- *单跳，基于基础设施*。这些网络具有与较大的有线网络（如因特网）连接的基站。此外，该基站与无线主机之间的所有通信都经过一个无线跳。你在教室、咖啡屋或图书馆中所使用的802.11网络，以及我们将很快学习的4G LTE数据网络都属于这种类型。我们日常的绝大部分时间是在与单跳、基于基础设施的无线网络打交道。
- *单跳，无基础设施*。在这些网络中，不存在与无线网络相连的基站。然而，如我们将要见到的那样，在这种单跳网络中的一个节点可以协调其他节点的传输。蓝牙网络（该网络连接诸如键盘、扬声器和戴在头上的耳机等小型无线设备，我们将在7.3.6节中学习）和具有自组织模式的802.11网络是单跳、无基础设施的网络。
- *多跳，基于基础设施*。在这些网络中，一个基站表现为以有线方式与较大网络相

连。然而，某种无线节点为了经该基站通信，可能不得不通过其他无线节点中继它们的通信。某些无线传感网络和所谓**无线网状网络**（wireless mesh network）就属于这种类型。

- *多跳，无基础设施*。在这些网络中没有基站，并且节点为了到达目的地可能必须在几个其他无线节点之间中继报文。节点也可能是移动的，在多个节点中改变连接关系，一类网络被称为**移动自组织网络**（mobile ad hoc network，MANET）。如果该移动节点是车载的，该网络是**车载自组织网络**（vehicular ad hoc network，VANET）。如你可能想象的那样，为这种网络开发协议是一种挑战，这是许多进行中的研究主题。

在本章中，我们将把主要学习内容限制在单跳网络，并且大多数是基于基础设施的网络。

现在我们更深一步地研究无线网络和移动网络面对的挑战。我们将首先讨论单独的无线链路，而在本章稍后部分讨论移动性。

历史事件

公共 WiFi 接入：路灯方案很快将实现吗？

WiFi 热区（即用户能够找到 802.11 无线接入的公共位置）在全世界的旅馆、机场和咖啡屋变得日益常见。大多数大学校园提供了无所不在的无线接入，找一个不提供无线因特网接入的旅馆也很难。

在过去的十年中，一些城市已经设计、部署和运行了市政的 WiFi 网络。作为公共服务（非常像街灯）提供无所不在的 WiFi 接入到社区的前景引人入胜，通过向所有居民提供因特网接入，有助于填平数字鸿沟，并能促进经济发展。世界上的许多城市，包括费城、香港、明尼阿波利斯、伦敦和奥克兰，已经计划在城市中提供无所不在的无线，或者已经在不同程度上做到了这一点。美国宾夕法尼亚州费城的目标是"将费城变为国家最大的 WiFi 热区，有助于改善教育，填平数字鸿沟，增强邻居交往，以及减少政府成本"。这项雄心勃勃的规划依据该城市、无线费城（一个非营利组织）和因特网服务提供商 Earthlink 达成的一项协议，这将在覆盖城市 80% 区域的街道路灯杆和交通控制设备上建设一个 802.11b 热区的运行网络。而由该网络产生的金融和运行股份在 2008 年出售给了一群私人投资者，他们在 2010 年再将该网络回售给该城市。其他城市如明尼阿波利斯、多伦多、香港、伦敦和奥克兰已经取得了较小规模的成功。

运行在无须许可证的频谱上（因此能够在无须购买昂贵的频谱使用权的情况下部署）的 802.11 网络看起来使其在资金上具有吸引力。然而，802.11 接入点（参见 7.3 节）比起 4G 蜂窝基站（参见 7.4 节）缺少大量的频段，要求部署大量的端点来覆盖相同的地理区域。在另一方面，提供因特网接入的蜂窝数据网络运行在许可证频谱上。蜂窝提供商为其网络的频谱接入权支付数十亿美元，使得蜂窝数据网络是商务而不是市政事务。

7.2 无线链路和网络特征

我们开始考虑用一台有线以太网交换机互联主机的一个简单有线网络，如一个家庭网

络（参见6.4节）。如果我们用无线802.11网络代替该有线以太网，用无线网络接口代替主机的有线以太网接口，用接入点代替以太网交换机，但实际上在网络层及其以上层次中不需要有任何变化。这提示我们当寻找有线和无线网络的重要区别时，应该重点关注链路层。我们的确能够发现有线链路和无线链路间的许多重要区别：

- 递减的信号强度。电磁波在穿过物体（如无线电信号穿过墙壁）时强度将减弱。即使在自由空间中，信号仍将扩散，这使得信号强度随着发送方和接收方距离的增加而减弱（有时称其为**路径损耗**（path loss））。
- 来自其他源的干扰。在同一个频段发送信号的电波源将相互干扰。例如，2.4GHz无线电话和802.11b无线LAN在相同的频段中传输。因此，802.11b无线LAN用户若同时利用2.4GHz无线电话通信，将会导致网络和电话都不会工作得特别好。除了来自发送源的干扰，环境中的电磁噪声（如附近的电动机、微波）也能形成干扰。
- 多径传播。当电磁波的一部分受物体和地面反射，在发送方和接收方之间走了不同长度的路径，则会出现**多径传播**（multipath propagation）。这使得接收方收到的信号变得模糊。位于发送方和接收方之间的移动物体可导致多径传播随时间而改变。

对于无线信道特征、模型和测量的详细讨论请参见［Anderson 1995］。

上述讨论表明，无线链路中的比特差错将比有线链路中更为常见。因此，无线链路协议（如我们将在下面一节中讨论的802.11协议）不仅采用有效的CRC错误检测码，还采用了链路层ARQ协议来重传受损的帧。

考虑了在无线信道上可能出现的损伤后，我们将注意力转向接收无线信号的主机。该主机接收到一个电磁信号，而该信号是发送方传输的初始信号的退化形式和环境中的背景噪声的结合，其中的信号退化是由于衰减和我们前面讨论过的多路径传播以及其他一些因素所引起的。**信噪比**（Signal-to-Noise Ratio，SNR）是所收到的信号（如被传输的信息）和噪声强度的相对测量。SNR的度量单位通常是分贝（dB），有人认为这个主要由电气工程师所使用的度量单位会使计算机科学家迷惑不解。以dB度量的SNR是下列比值的20倍，即接收到的信号的振幅与噪声的振幅的以10为底的对数的比值。就我们的讨论目的而言，我们仅需要知道较大的SNR使接收方更容易从背景噪声中提取传输的信号。

图7-3（该图选自［Holland 2001］）显示了三种不同的调制技术的**比特差错率**（BER）（大致说来，BER是在接收方收到的有错传输比特的概率）与SNR之比，这些调制技术用于对信息进行编码以在理想信道上传输。调制和编码理论以及信号提取和BER都超出了本书的范围（对这些主题的讨论参见［Schwart 1980］）。尽管如此，图7-3显示了几种物理层的特征，这些特征对于理解较高层无线通信协议是重要的：

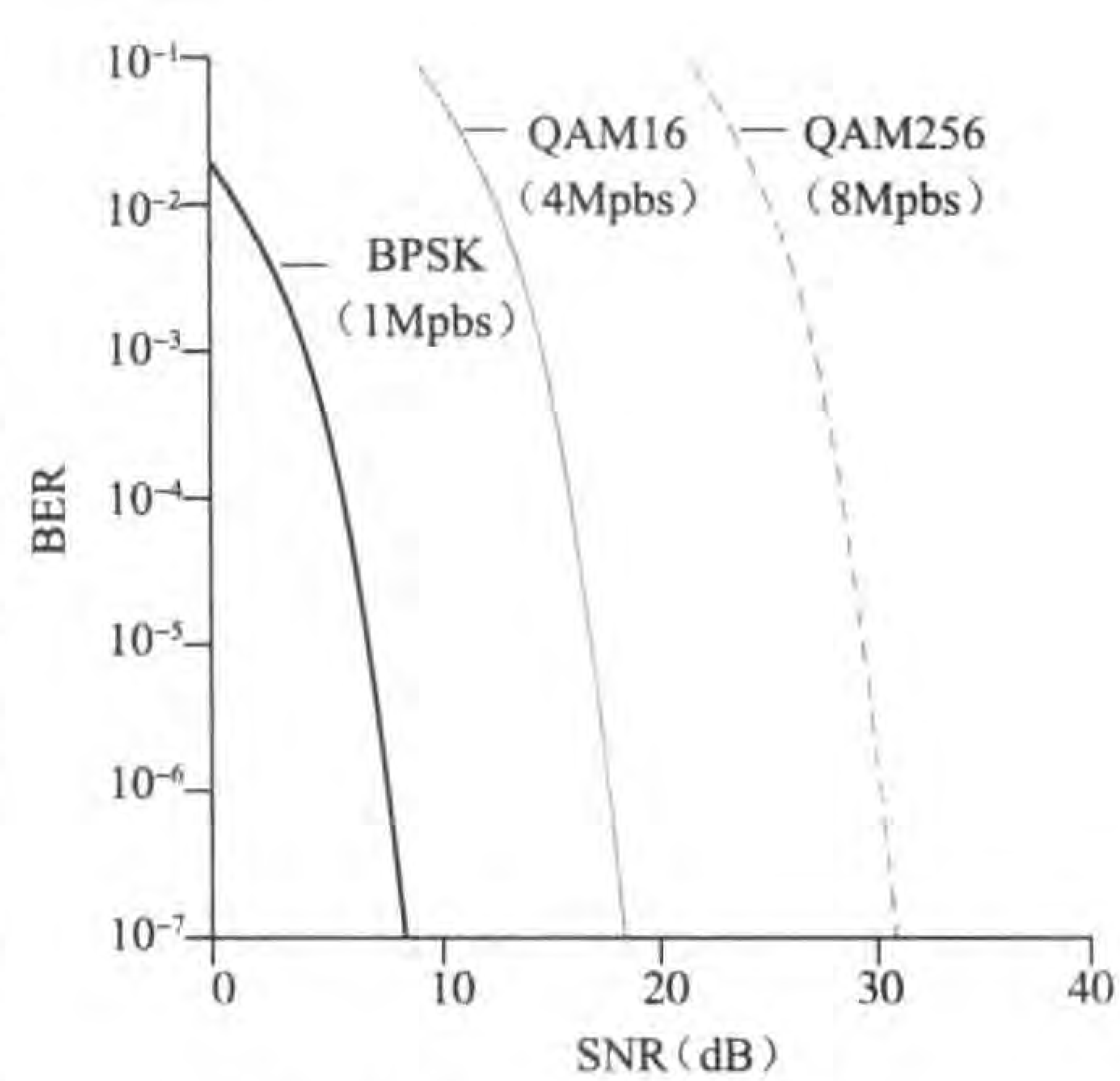


图7-3　比特差错率、传输率和SNR

- 对于给定的调制方案，SNR越高，BER越低。由于发送方通过增加它的传输功率就能够增加SNR，因此发送方能够通

过增加它的传输功率来降低接收到差错帧的概率。然而，注意到当该功率超过某个阈值时，如 BER 从 10^{-12}降低到 10^{-13}，可证明几乎不会有实际增益。增加传输功率也会伴随着一些缺点：发送方必须消耗更多的能量（对于用电池供电的移动用户，这一点非常重要），并且发送方的传输更可能干扰另一个发送方的传输（参见图 7-4b）。

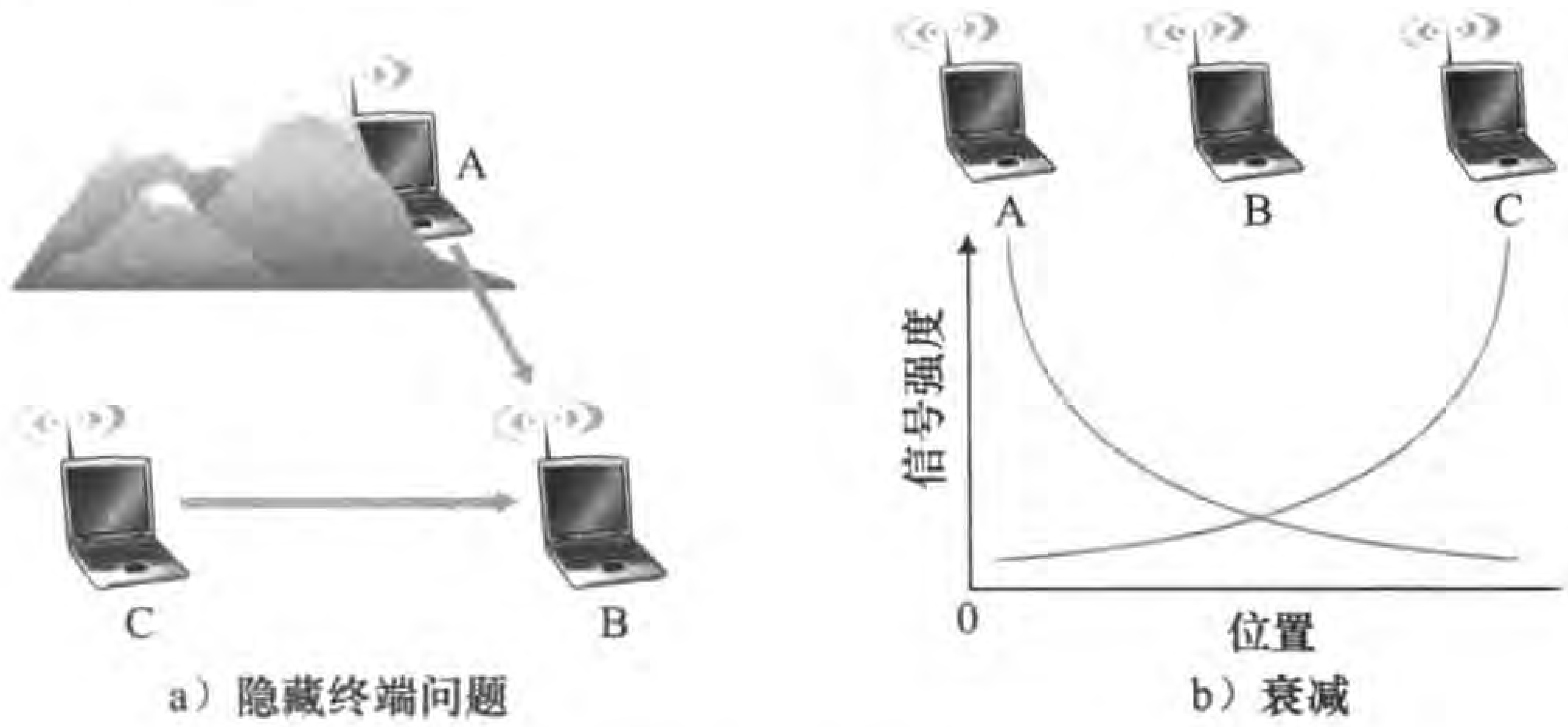


图 7-4　隐藏终端问题和衰减

- **对于给定的 SNR，具有较高比特传输率的调制技术（无论差错与否）将具有较高的 BER。**例如在图 7-3 中，对于 10dB 的 SNR，具有 1Mbps 传输速率的 BPSK 调制具有小于 10^{-7}的 BER，而具有 4Mbps 传输速率的 QAM 16 调制，BER 是 10^{-1}，该值太高而没有实际用处。然而，具有 20dB 的 SNR，QAM 16 调制具有 4Mbps 的传输速率和 10^{-7}的 BER，而 BPSK 调制具有仅 1Mbps 的传输速率和一个低得“无法在图上表示”的 BER。如果人们能够容忍 10^{-7}的 BER，在这种情况下由 QAM 16 提供的较高的传输速率将使它成为首选的调制技术。这些考虑引出了我们下面描述的最后一个特征。
- **物理层调制技术的动态选择能用于适配对信道条件的调制技术。**SNR（因此 BER）可能因移动性或由于环境中的改变而变化。在蜂窝数据系统中以及在 802.11 WiFi 和 4G 蜂窝数据网络中（我们将在 7.3 节和 7.4 节中学习）使用了自适应调制和编码。例如，这使得对于给定的信道特征选择一种调制技术，在受制于 BER 约束的前提下提供最高的可能传输速率。

有线和无线链路之间的差异并非仅仅只有较高的、时变的误比特率这一项。前面讲过在有线广播链路中所有节点能够接收到所有其他节点的传输。而在无线链路中，情况并非如此简单。如图 7-4 所示，假设站点 A 正在向站点 B 发送，同时假定站点 C 也在向站点 B 传输。由于所谓的**隐藏终端问题**（hidden terminal problem），即使 A 和 C 的传输确实是在目的地 B 发生干扰，环境的物理阻挡（例如，一座大山或者一座建筑）也可能会妨碍 A 和 C 互相听到对方的传输。这种情况如图 7-4a 所示。第二种导致在接收方无法检测的碰撞情况是，当通过无线媒体传播时信号强度的**衰减**（fading）。图 7-4b 图示了这种情况，A 和 C 所处的位置使得它们的信号强度不足以使它们相互检测到对方的传输，然而它们的传输足以强到在站点 B 处相互干扰。正如我们将在 7.3 节看到的那样，隐藏终端问题和衰减使得多路访问在无线网络中的复杂性远高于在有线网络中的情况。

CDMA

在第6章讲过，当不同主机使用一个共享媒体通信时，需要有一个协议来保证多个发送方发送的信号不在接收方互相干扰。在第6章中，我们描述了3类媒体访问协议：信道划分、随机访问和轮流。码分多址（Code Division Multiple Access，CDMA）属于信道划分协议族。它在无线LAN和蜂窝技术中应用很广泛。由于CDMA对无线领域十分重要，在后面小节中对具体的无线接入技术进行探讨以前，我们首先对其快速地浏览一下。

在CDMA协议中，要发送的每个比特都通过乘以一个信号（编码）的比特来进行编码，这个信号的变化速率（通常称为**码片速率**，chipping rate）比初始数据比特序列的变化速率快得多。图7-5表示一个简单的、理想化的CDMA编码/解码情形。假设初始数据比特到达CDMA编码器的速率定义了时间单元；也就是说，每个要发送的初始数据比特需要1比特时隙时间。设 d_i 为第 i 个比特时隙中的数据比特值。为了数学上便利，我们把具有0值的数据比特表示为 -1。每个比特时隙又进一步细分为 M 个微时隙；在图7-5中，$M=8$，不过在实际中 M 的值要大得多。发送方使用的CDMA编码由 M 个值的一个序列 c_m 组成，$m=1$，…，M，每个取值为 $+1$ 或者 -1。在图7-5的例子中，被发送方使用的 M 比特的CDMA码是（1，1，1，-1，1，-1，-1，-1）。

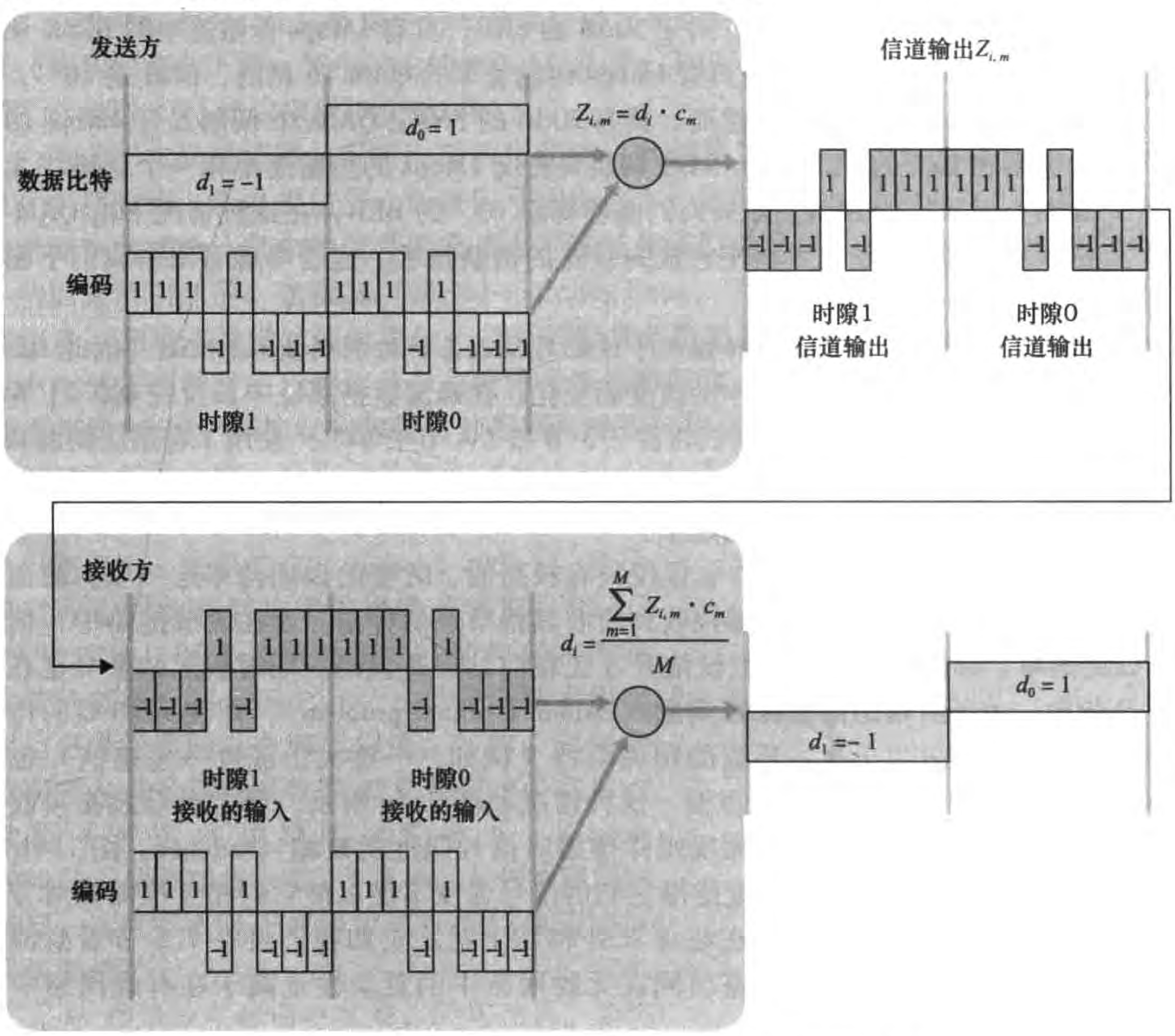


图7-5 一个简单的CDMA例子：发送方编码，接收方解码

为了说明 CDMA 的工作原理，我们关注第 i 个数据比特 d_i。对于 d_i 比特传输时间的第 m 个微时隙，CDMA 编码器的输出 $Z_{i,m}$ 是 d_i 乘以分配的 CDMA 编码的第 m 比特 c_m：

$$Z_{i,m} = d_i \cdot c_m \tag{7-1}$$

简单地说，对没有干扰的发送方，接收方将收到编码的比特 $Z_{i,m}$，并且恢复初始的数据比特 d_i，计算如下：

$$d_i = \frac{1}{M}\sum_{m=1}^{M} Z_{i,m} \cdot c_m \tag{7-2}$$

读者可能想通过推敲图 7-5 所示例子的细节，来明白使用式（7-2）在接收方确实正确恢复了初始数据比特。

然而，这个世界远不是理想化的，如上面所述，CDMA 必须在存在干扰发送方的情况下工作，这些发送方用分配的不同编码来编码和传输它们的数据。但是当一个发送方的数据比特和其他发送方发送的比特混在一起时，一个 CDMA 接收方怎样恢复该发送方的初始数据比特呢？CDMA 的工作有一种假设，即对干扰的传输比特信号是加性的，这意味着，例如在同一个微时隙中，如果 3 个发送端都发送 1，第 4 个发送端发送 -1，那么在那个微时隙中所有的接收方接收的信号都是 2（因为 1+1+1-1=2）。在存在多个发送方时，发送方 s 计算它编码后的传输 $Z^s_{i,m}$，计算方式与式（7-1）中的完全相同。然而在第 i 个比特时隙的第 m 个微时隙期间，接收方现在收到的值是在那个微时隙中从所有 N 个发送方传输的比特的总和：

$$Z^*_{i,m} = \sum_{s=1}^{N} Z^s_{i,m}$$

令人吃惊的是，如果仔细地选择发送方的编码，每个接收方只通过式（7-2）中的同样的方式使用发送方的编码，就能够从聚合的信号中恢复一个给定的发送方发送的数据：

$$d_i = \frac{1}{M}\sum_{m=1}^{M} Z^*_{i,m} \cdot c_m \tag{7-3}$$

如在图 7-6 中所示，描述了两个发送方的 CDMA 例子。上部的发送方使用的 M 比特 CDMA 编码是（1，1，1，-1，1，-1，-1，-1），而下部的发送方使用的 CDMA 编码是（1，-1，1，1，1，-1，1，1）。图 7-6 描述了一个接收方恢复从上部发送方发送的初始数据比特的情况。注意到这个接收方能够提取来自发送方 1 的数据，而不管来自发送方 2 的干扰传输。

再回到我们第 6 章中鸡尾酒会的类比，一个 CDMA 协议类似于让聚会客人使用多种语言来谈论；在这种情况下，人们实际上非常善于锁定他们能听懂的语言的谈话，而过滤了其余的谈话。我们这里看到 CDMA 是一个划分协议，因为它划分编码空间（与时间或频率相对），并且给每个节点分配一段专用的代码空间。

我们这里对 CDMA 的讨论是简要的；实践中还必须处理大量的困难问题。首先，为了使 CDMA 接收方能够提取一个特定的发送方的信号，必须仔细地选择 CDMA 编码。其次，我们的讨论假设在接收方接收到的来自从不同发送方的信号强度是相同的；这可能在实际中很难获得。有大量的文章讨论了有关 CDMA 的这些和其他问题；详细内容见［Pickholtz 1982；Viterbi 1995］。

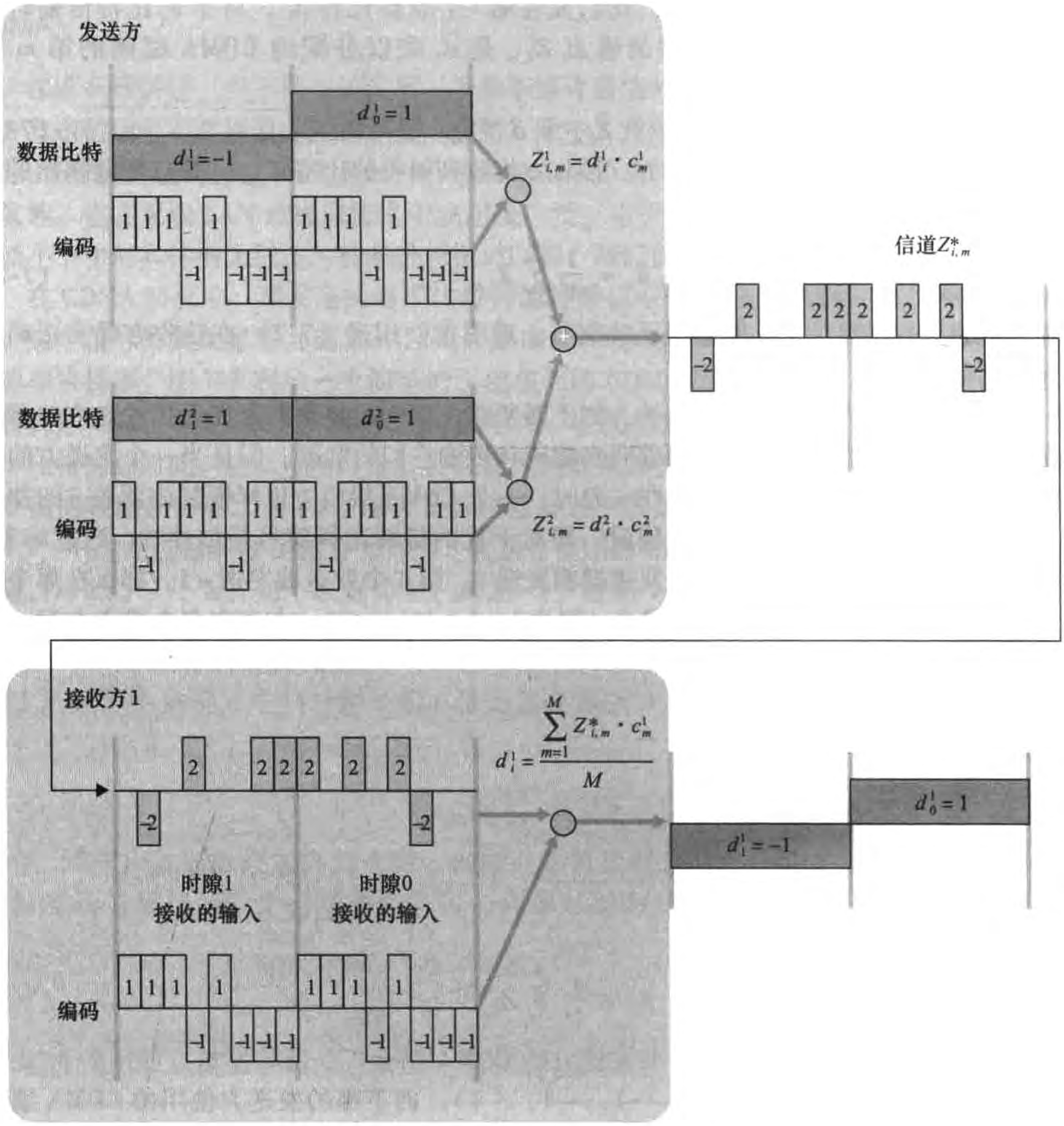


图 7-6 两个发送方的 CDMA 例子

7.3 WiFi：802.11 无线 LAN

当前，无线 LAN 在工作场所、家庭、教育机构、咖啡屋、机场以及街头无所不在，它已经成为因特网中的一种十分重要的接入网技术。尽管在 20 世纪 90 年代研发了许多有关无线 LAN 的标准和技术，但其中有一类标准已经明显成为赢家：IEEE 802.11 无线 LAN（也称为 WiFi）。在本节中，我们将详细研究 802.11 无线 LAN，分析它的帧结构、它的媒体访问协议以及 802.11 LAN 与有线以太网 LAN 的互联。

在 IEEE 802.11（“WiFi”）协议族中有几套有关无线 LAN 的 802.11 标准，表 7-1 中对它们进行了总结。不同的 802.11 标准都具有某些共同的特征。它们都使用相同的媒体访问协议 CSMA/CA，我们稍后将讨论该协议。这些标准对它们的链路层帧也都使用相同

的帧结构，而且它们都具有减少其传输速率的能力以伸展到更远的距离。并且重要的是，802.11 产品也都是向后兼容的，这意味着例如仅支持 802.11g 的移动站仍可以与较新的 802.11ac 基站交互。

然而，如表 7-1 所示，这些标准在物理层有一些重要的区别。802.11 设备工作在两个不同的频率段上：2.4～2.485GHz（称之为 2.4GHz 频段）和 5.1～5.8GHz（称之为 5GHz 频段）。2.4GHz 频段是一种无须执照的频段，在此频段上，使用 2.4GHz 的电话和微波炉等 802.11 设备可能会争用该频段的频谱。在 5GHz 频段，对于给定的功率等级 802.11 LAN 有更短的传输距离，并且受多径传播的影响更多。两种最新的标准 802.11n [IEEE 802.11n 2012] 和 802.11ac [IEEE 802.11ac 2013; Cisco 802.11ac 2015] 使用多输入多输出（MIMO）天线；也就是说在发送一侧的两根或更多的天线以及在接收一侧的两根或更多的天线发送/接收着不同的信号 [Diggavi 2004]。802.11ac 基站可以同时向多个站点传输，并且使用“智能”天线在接收方的方向上用自适应成型波束向目标传输。这减少了干扰并增大了以给定数据率传输的可达距离。在表 7-1 中显示的数据率是针对理想环境的数据，例如一个离基站 1 米远的接收方没有干扰，而这种场景在实践中是不可能经历到的！因此正如谚语所说：你走过的路（或者此时是你的无线数据率）也许是变化的（YMMV）。

表 7-1 IEEE 802.11 标准小结

标准	频率范围	数据率
802.11b	2.4GHz	最高为 11Mbps
802.11a	5GHz	最高为 54Mbps
802.11g	2.4GHz	最高为 54Mbps
802.11n	2.5～5GHz	最高为 450Mbps
802.11ac	5GHz	最高为 1300Mbps

7.3.1 802.11 体系结构

图 7-7 显示了 802.11 无线 LAN 体系结构的基本构件。802.11 体系结构的基本构件模块是**基本服务集**（Basic Service Set，BSS）。一个 BSS 包含一个或多个无线站点和一个在 802.11 术语中称为**接入点**（Access Point，AP）的中央**基站**（base station）。图 7-7 展示了两个 BSS 中的 AP，它们连接到一个互联设备上（如交换机或者路由器），互联设备又连接到因特网中。在一个典型的家庭网络中，有一个 AP 和一台将该 BSS 连接到因特网中的路由器（通常综合成为一个单元）。

与以太网设备类似，每个 802.11 无线站点都具有一个 6 字节的 MAC 地址，该地址存储在该站适配器（即 802.11 网络接口卡）的固件中。每个 AP 的无线接口也具有一个 MAC 地址。与以太网类似，这些 MAC 地址由 IEEE 管理，理论上是全球唯一的。

如 7.1 节所述，配置 AP 的无线 LAN 经常被称作**基础设施无线** LAN（infrastructure wireless LAN），其中的“基础设施”是指 AP 连同互联 AP 和一台路由器的有线以太网。图 7-8 显示了 IEEE 802.11 站点也能将它们自己组合在一起形成一个自组织网络，即一个无中心控制和与“外部世界”无连接的网络。这里，该网络是由彼此已经发现相互接近且有通信需求的移动设备“动态”形成，并且在它们所处环境中没有预先存在的网络基础设施。当携带便携机的人们聚集在一起时（例如，在一个会议室、一列火车或者一辆汽车中），并且要在没有中央化的 AP 的情况下交换数据，一个自组织网络就可能形成了。随着要通信的便携设备的继续激增，人们对自组织网络产生巨大的兴趣。然而在本节中，我们只关注基础设施无线 LAN。

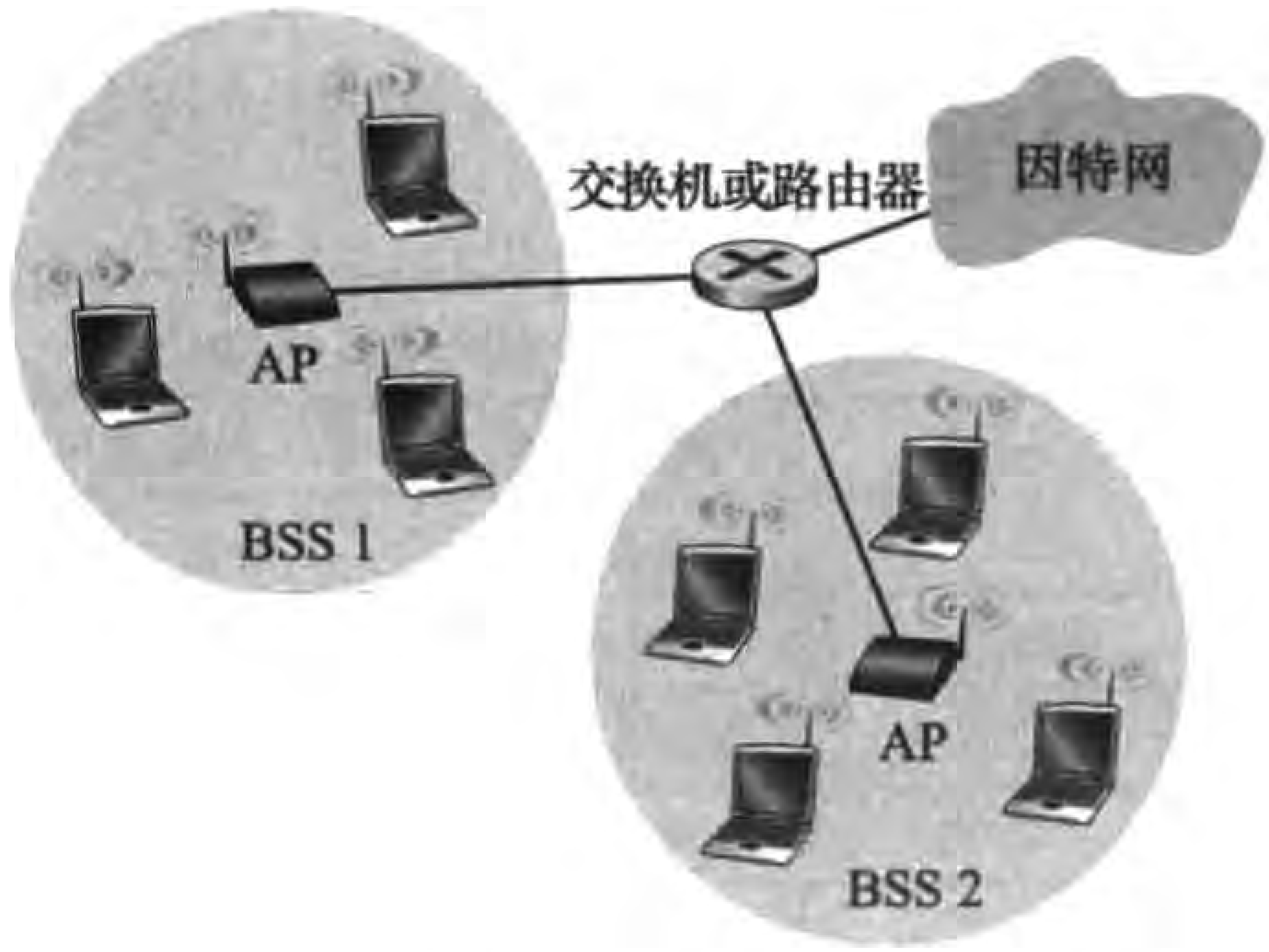


图 7-7 IEEE 802.11 LAN 体系结构



图 7-8 IEEE 802.11 自组织网络

信道与关联

在 802.11 中，每个无线站点在能够发送或者接收网络层数据之前，必须与一个 AP 相关联。尽管所有 802.11 标准都使用了关联，但我们将专门在 IEEE 802.11b/g 环境中讨论这一主题。

当网络管理员安装一个 AP 时，管理员为该接入点分配一个单字或双字的**服务集标识符**（Service Set Identifier，SSID）。（例如，当你在 iPhone 上选择设置 WiFi 时，将显示某范围内每个 AP 的 SSID。）管理员还必须为该 AP 分配一个信道号。为了理解信道号，回想前面讲过的 802.11 运行在 2.4 ~ 2.4835GHz 的频段中。在这个 85MHz 的频段内，802.11 定义了 11 个部分重叠的信道。当且仅当两个信道由 4 个或更多信道隔开时它们才无重叠。特别是信道 1、6 和 11 的集合是唯一的 3 个非重叠信道的集合。这意味着管理员可以在同一个物理网络中安装 3 个 802.11b AP，为这些 AP 分配信道 1、6 和 11，然后将每个 AP 都连接到一台交换机上。

既然已经对 802.11 信道有了基本了解，我们则可以描述一个有趣（且并非完全不寻常）的情况，即有关 WiFi 丛林。**WiFi 丛林**（WiFi jungle）是一个任意物理位置，在这里无线站点能从两个或多个 AP 中收到很强的信号。例如，在纽约城的许多咖啡馆中，无线站点可以从附近许多 AP 中选取一个信号。其中一个 AP 可能由该咖啡馆管理，而其他 AP 可能位于咖啡馆附近的住宅区内。这些 AP 中的每一个都可能位于不同的子网中，并被独立分配一个信道。

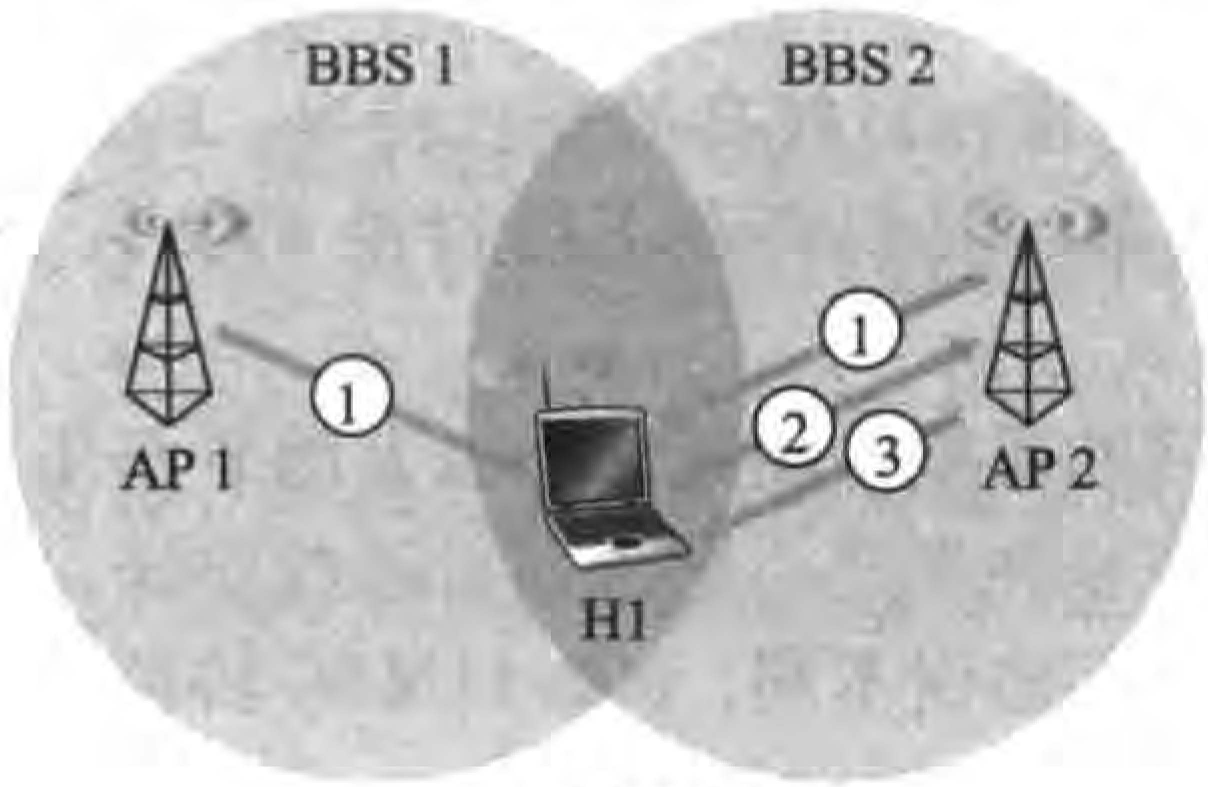
现在假定你带着自己的手机、平板电脑或便携机进入这样一个 WiFi 丛林，寻求无线因特网接入和一个蓝莓松饼。设在这个丛林中有 5 个 AP。为了获得因特网接入，你的无线站点需要加入其中一个子网并因此需要与其中的一个 AP 相**关联**（associate）。关联意味着这一无线站点在自身和该 AP 之间创建一个虚拟线路。特别是，仅有关联的 AP 才向你的无线站点发送数据帧，并且你的无线站点也仅仅通过该关联 AP 向因特网发送数据帧。然而，你的无线站点是如何与某个特定的 AP 相关联的？更为根本的问题是，你的无线站点是如何知道哪个 AP 位于该丛林呢？

802.11 标准要求每个 AP 周期性地发送**信标帧**（beacon frame），每个信标帧包括该 AP 的 SSID 和 MAC 地址。你的无线站点为了得知正在发送信标帧的 AP，扫描 11 个信道，找出来自可能位于该区域的 AP 所发出的信标帧（其中一些 AP 可能在相同的信道中传输，即这里有一个丛林！）。通过信标帧了解到可用 AP 后，你（或者你的无线主机）选择一个

AP 用于关联。

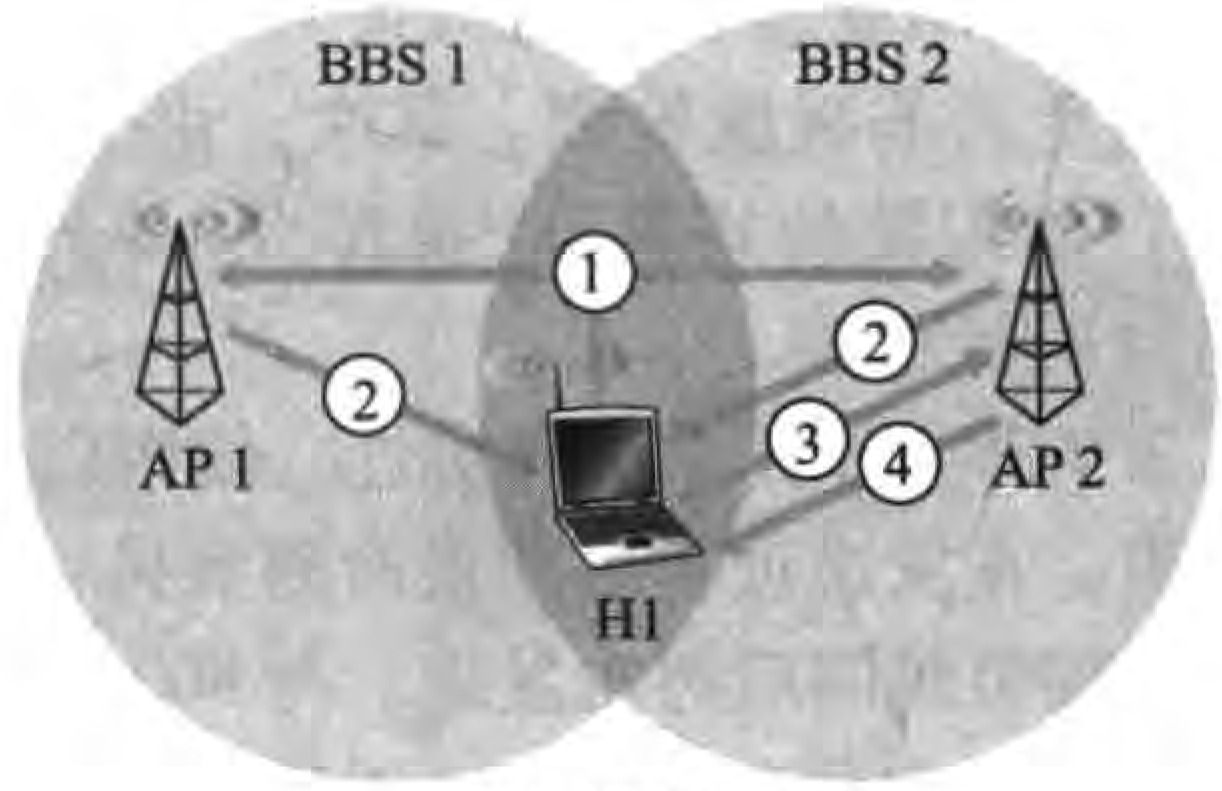
802.11 标准没有指定选择哪个可用的 AP 进行关联的算法；该算法被遗留给 802.11 固件和无线主机的软件设计者。通常，主机选择接收到的具有最高信号强度的信标帧。虽然高信号强度好（例如可参见图 7-3），信号强度将不是唯一决定主机接收性能的 AP 特性。特别是，所选择的 AP 可能具有强信号，但可能被其他附属的主机（将需要共享该 AP 的无线带宽）所过载，而某未过载的 AP 由于稍弱的信号而未被选择。选择 AP 的一些可替代的方法近来已被提出［Vasudevan 2005；Nicholson 2006；Sudaresan 2006］。有关信号强度如何测量的有趣而朴实的讨论参见［Bardwell 2004］。

扫描信道和监听信标帧的过程被称为**被动扫描**（passive scanning）（参见图 7-9a）。无线主机也能够执行**主动扫描**（active scanning），这是通过向位于无线主机范围内的所有 AP 广播探测帧完成的，如图 7-9b 所示。AP 用一个探测响应帧应答探测请求帧。无线主机则能够在响应的 AP 中选择某 AP 与之相关联。



a）被动扫描

①自AP发送信标帧
②H1向选择的AP发送关联请求帧
③选择的AP向H1发送关联响应帧



b）主动扫描

①自H1广播探测请求帧
②自AP发送探测响应
③H1项选择的AP发送关联请求帧
④选择的AP向H1发送关联的响应帧

图 7-9　对接入点的主动和被动扫描

选定与之关联的 AP 后，无线主机向 AP 发送一个关联请求帧，并且该 AP 以一个关联响应帧进行响应。注意到对于主动扫描需要这种第二次请求/响应握手，因为一个对初始探测请求帧进行响应的 AP 并不知道主机选择哪个（可能多个）响应的 AP 进行关联，这与 DHCP 客户能够从多个 DHCP 服务器进行选择有诸多相同之处（参见图 4-24）。一旦与一个 AP 关联，该主机希望加入该 AP 所属的子网中（以 4.3.3 节中的 IP 寻址的意义）。因此。该主机通常将通过关联的 AP 向该子网发送一个 DHCP 发现报文（参见图 4-24），以获取在该 AP 子网中的一个 IP 地址。一旦获得地址，网络的其他部分将直接视你的主机为该子网中的另一台主机。

为了与特定的 AP 创建一个关联，某无线站点可能要向该 AP 鉴别它自身。802.11 无线 LAN 提供了几种不同的鉴别和接入方法。一种被许多公司采用的方法是，基于一个站点的 MAC 地址允许其接入一个无线网络。第二种被许多因特网咖啡屋采用的方法是，应用用户名和口令。在两种情况下，AP 通常与一个鉴别服务器进行通信，使用一种诸如 RADIUS［RFC 2865］或 DIAMETER［RFC 3588］的协议，在无线终端站和鉴别服务器之间中继信息。分离鉴别服务器和 AP，使得一个鉴别服务器可以服务于多个 AP，将（经常是敏感的）鉴别和接入的决定集中到单一服务器中，使 AP 费用和复杂性较低。我们将在第 8 章看到，定义 802.11 协议族安全性的新 IEEE 802.11i 协议就恰好

采用了这一方法。

7.3.2 802.11 MAC 协议

一旦某无线站点与一个 AP 相关联，它就可以经该接入点开始发送和接收数据帧。然而因为许多无线设备或 AP 自身可能希望同时经过相同信道传输数据帧，因此需要一个多路访问协议来协调传输。下面，我们将无线设备或 AP 称为**站点**（station），它们共享多个接入信道。正如在第 6 章和 7.2.1 节中讨论的那样，宽泛地讲有三类多路访问协议：信道划分（包括 CDMA）、随机访问和轮流。受以太网及其随机访问协议巨大成功的激励，802.11 的设计者为 802.11 无线 LAN 选择了一种随机访问协议。这个随机访问协议称作**带碰撞避免的 CSMA**（CSMA with collision avoidance），或简称为 **CSMA/CA**。与以太网的 CSMA/CD 相似，CSMA/CA 中的“CSMA”代表“载波侦听多路访问”，意味着每个站点在传输之前侦听信道，并且一旦侦听到该信道忙则抑制传输。尽管以太网和 802.11 都使用载波侦听随机接入，但这两种 MAC 协议有重要的区别。首先，802.11 使用碰撞避免而非碰撞检测。其次，由于无线信道相对较高的误比特率，802.11（不同于以太网）使用链路层确认/重传（ARQ）方案。我们将在下面讨论 802.11 的碰撞避免和链路层确认机制。

在 6.3.2 节和 6.4.2 节曾讲过，使用以太网的碰撞检测算法，以太网节点在发送过程中监听信道。在发送过程中如果检测到另一节点也在发送，则放弃自己的发送，并且在等待一个小的随机时间后再次发送。与 802.3 以太网协议不同，802.11MAC 协议并未实现碰撞检测。这主要由两个重要的原因所致：

- 检测碰撞的能力要求站点具有同时发送（站点自己的信号）和接收（检测其他站点是否也在发送）的能力。因为在 802.11 适配器上，接收信号的强度通常远远小于发送信号的强度，制造具有检测碰撞能力的硬件代价较大。
- 更重要的是，即使适配器可以同时发送和监听信号（并且假设它一旦侦听到信道忙就放弃发送），适配器也会由于隐藏终端问题和衰减问题而无法检测到所有的碰撞，参见 7.2 节的讨论。

由于 802.11 无线局域网不使用碰撞检测，一旦站点开始发送一个帧，它就完全地发送该帧；也就是说，一旦站点开始发送，就不会返回。正如人们可能猜想的那样，碰撞存在时仍发送整个数据帧（尤其是长数据帧）将严重降低多路访问协议的性能。为了降低碰撞的可能性，802.11 采用几种碰撞避免技术，我们稍后讨论它们。

然而，在考虑碰撞避免之前，我们首先需要分析 802.11 的**链路层确认**（link-layer acknowledgment）方案。7.2 节讲过，当无线 LAN 中某站点发送一个帧时，该帧会由于多种原因不能无损地到达目的站点。为了处理这种不可忽视的故障情况，802.11 MAC 使用链路层确认。如图 7-10 所示，目的站点收到一个通过 CRC 校验的帧后，它等待一个被称作**短帧间间隔**（Short Inter-Frame Spacing，SIFS）的一小段时间，然后发回一个确认帧。如果发送站点在给定的时间内未收到确认帧，它假定出现了错误并重传该帧，使用 CSMA/CA 协议访问该信道。如果在若干固定次重传后仍未收到确认，发送站点将放弃发送并丢弃该帧。

讨论过 802.11 如何使用链路层确认后，我们可以描述 802.11 的 CSMA/CA 协议了。假设一个站点（无线站点或者 AP）有一个帧要发送。

1）如果某站点最初监听到信道空闲，它将在一个被称作**分布式帧间间隔**（Distributed

Inter-Frame Space，DIFS）的短时间段后发送该帧，如图 7-10 所示。

2）否则，该站点选取一个随机回退值（如我们在 6.3.2 节中遇到的那样）并且在侦听信道空闲时递减该值。当侦听到信道忙时，计数值保持不变。

3）当计数值减为 0 时（注意到这只可能发生在信道被侦听为空闲时），该站点发送整个数据帧并等待确认。

4）如果收到确认，发送站点知道它的帧已被目的站正确接收了。如果该站点要发送另一帧，它将从第二步开始 CSMA/CA 协议。如果未收到确认，发送站点将重新进入第二步中的回退阶段，并从一个更大的范围内选取随机值。

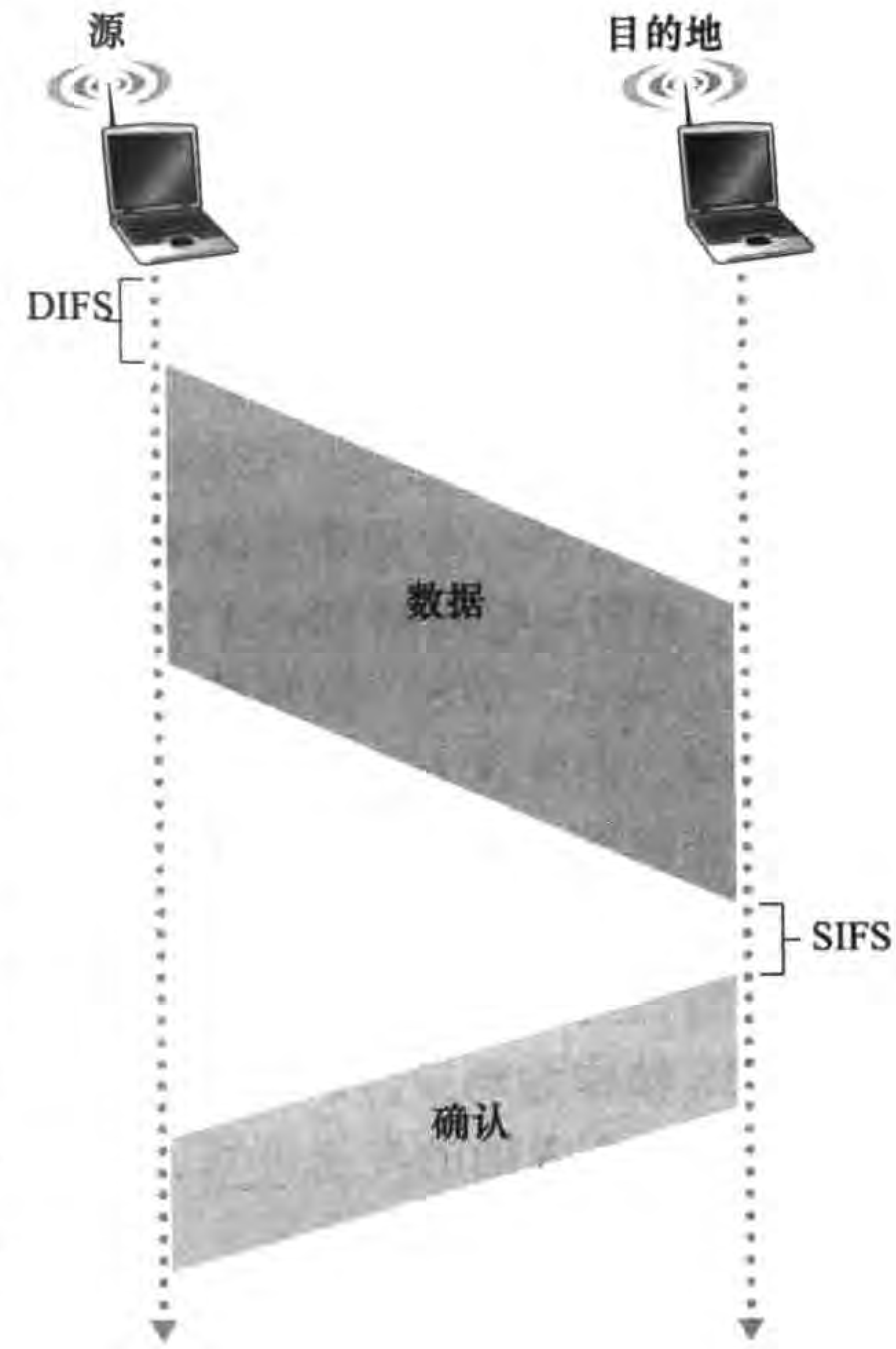


图 7-10　802.11 使用链路层确认

前面讲过，在以太网的 CSMA/CD 的多路访问协议（6.3.2 节）下，一旦侦听到信道空闲，站点开始发送。然而，使用 CSMA/CA，该站点在倒计数时抑制传输，即使它侦听到该信道空闲也是如此。为什么 CSMA/CD 和 CSMA/CA 采用了不同的方法呢？

为了回答这一问题，我们首先考虑这样一种情形：两个站点分别有一个数据帧要发送，但是，由于侦听到第三个站点已经在传输，双方都未立即发送。使用以太网的 CSMA/CD 协议中，两个站点将会在检测到第三方发送完毕后立即开始发送。这将导致一个碰撞，在 CSMA/CD 协议中碰撞并非是一个严重的问题，因为两个站点检测到碰撞后都会放弃它们的发送，从而避免了由于碰撞而造成的该帧剩余部分的无用发送。而在 802.11 中情况却十分不同，因为 802.11 并不检测碰撞和放弃发送，遭受碰撞的帧仍将被完全传输。因此 802.11 的目标是无论如何尽可能避免碰撞。在 802.11 中，如果两个站点侦听到信道忙，它们都将立即进入随机回退，希望选择一个不同的回退值。如果这些值的确不同，一旦信道空闲，其中的一个站点将在另一个之前发送，并且（如果两个站点均未对对方隐藏）“失败站点”将会听到“胜利站点”的信号，冻结它的计数器，并在胜利站点完成传输之前一直抑制传输。通过这种方式，避免了高代价的碰撞。当然，在以下情况下使用 802.11 仍可能出现碰撞：两个站点可能互相是隐藏的，或者两者可能选择了非常靠近的随机回退值，使来自先开始站点的传输也必须到达第二个站点。回想前面我们在图 6-12 的环境中讨论随机访问算法时遇到过这个问题。

1. 处理隐藏终端：RTS 和 CTS

802.11 MAC 协议也包括了一个极好（但为可选项）的预约方案，以帮助在出现隐藏终端的情况下避免碰撞。我们在图 7-11 的环境下研究这种方案，其中显示了两个无线站点和一个接入点。

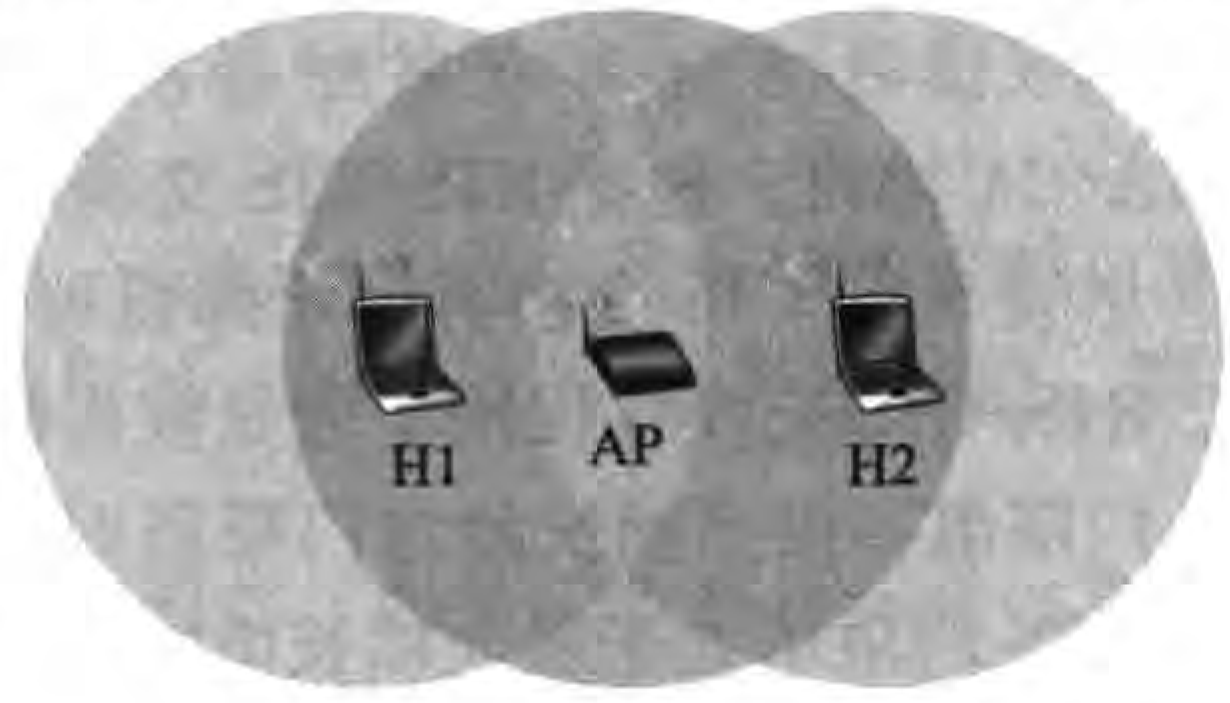


图 7-11　隐藏终端的例子：H1 和 H2 彼此互相隐藏

这两个无线站点都在该 AP 的覆盖范围内（其覆盖范围显示为阴影圆环），并且两者都与该 AP 相关联。然而，由于衰减，无线节点的信号范围局限在图 7-11 所示的阴影圆环内部。因此，尽管每个无线站点对 AP 都不隐藏，两者彼此却是隐藏的。

现在我们考虑为什么隐藏终端会导致出现问题。假设站点 H1 正在传输一个帧，并且在 H1 传输的中途，站点 H2 要向 AP 发送一个帧。由于 H2 未听到来自 H1 的传输，它将首先等待一个 DIFS 间隔，然后发送该帧，导致产生了一个碰撞。从而在 H1 和 H2 的整个发送阶段，信道都被浪费了。

为了避免这一问题，IEEE 802.11 协议允许站点使用一个短**请求发送**（Request to Send，RTS）控制帧和一个短**允许发送**（Clear to Send，CTS）控制帧来预约对信道的访问。当发送方要发送一个 DATA 帧时，它能够首先向 AP 发送一个 RTS 帧，指示传输 DATA 帧和确认（ACK）帧需要的总时间。当 AP 收到 RTS 帧后，它广播一个 CTS 帧作为响应。该 CTS 帧有两个目的：给发送方明确的发送许可，也指示其他站点在预约期内不要发送。

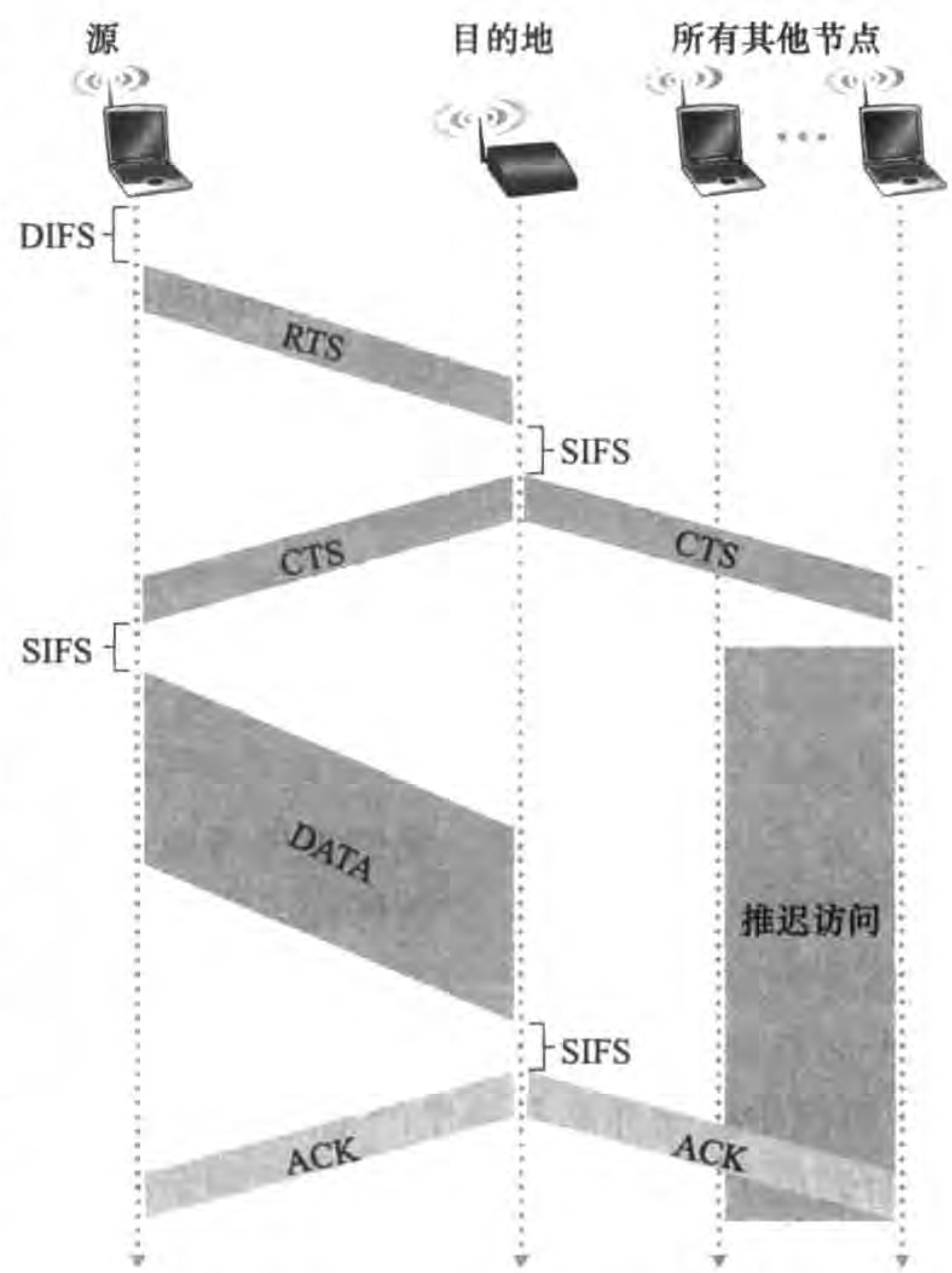


图 7-12　使用 RTS 和 CTS 帧的碰撞避免

因此，在图 7-12 中，在传输 DATA 帧前，H1 首先广播一个 RTS 帧，该帧能被其范围内包括 AP 在内的所有站点听到。AP 然后用一个 CTS 帧响应，该帧也被其范围内包括 H1 和 H2 在内的所有站点听到。站点 H2 听到 CTS 后，在 CTS 帧中指明的时间内将抑制发送。RTS、CTS、DATA和 ACK 帧如图 7-12 所示。

RTS 和 CTS 帧的使用能够在两个重要方面提高性能：

- 隐藏终端问题被缓解了，因为长 DATA 帧只有在信道预约后才被传输。
- 因为 RTS 和 CTS 帧较短，涉及 RTS 和 CTS 帧的碰撞将仅持续短 RTS 和 CTS 帧的持续期。一旦 RTS 和 CTS 帧被正确传输，后续的 DATA 和 ACK 帧应当能无碰撞地发送。

建议读者去查看本书配套网站上的 802.11 Java 程序。这个交互式程序演示了 CSMA/CA 协议，包括 RTS/CTS 交换序列。

尽管 RTS/CTS 交换有助于降低碰撞，但它同样引入了时延以及消耗了信道资源。因此，RTS/CTS 交换仅仅用于为长数据帧预约信道。在实际中，每个无线站点可以设置一个 RTS 门限值，仅当帧长超过门限值时，才使用 RTS/CTS 序列。对许多无线站点而言，默认的 RTS 门限值大于最大帧长值，因此对所有发送的 DATA 帧，RTS/CTS 序列都被跳过。

2. 使用 802.11 作为一个点对点链路

到目前为止我们的讨论关注在多路访问环境中使用 802.11。应该指出，如果两个节点

每个都具有一个定向天线，它们可以将其定向天线指向对方，并基本上是在一个点对点的链路上运行 802.11 协议。如果商用 802.11 硬件产品价格低廉，那么使用定向天线以及增加传输功率使得 802.11 成为一个在数十公里距离中提供无线点对点连接的廉价手段。[Raman 2007] 描述了这样一个运行于印度恒河郊区平原上的多跳无线网络，其中包含了点对点 802.11 链路。

7.3.3 IEEE 802.11 帧

尽管 802.11 帧与以太网帧有许多共同特点，但它也包括了许多特定用于无线链路的字段。8021.11 帧如图 7-13 所示，在该帧上的每个字段上面的数字代表该字段以字节计的长度；在该帧控制字段中，每个子字段上面的数字代表该子字段以比特计的长度。现在我们查看该帧中各字段以及帧控制字段中一些重要的子字段。

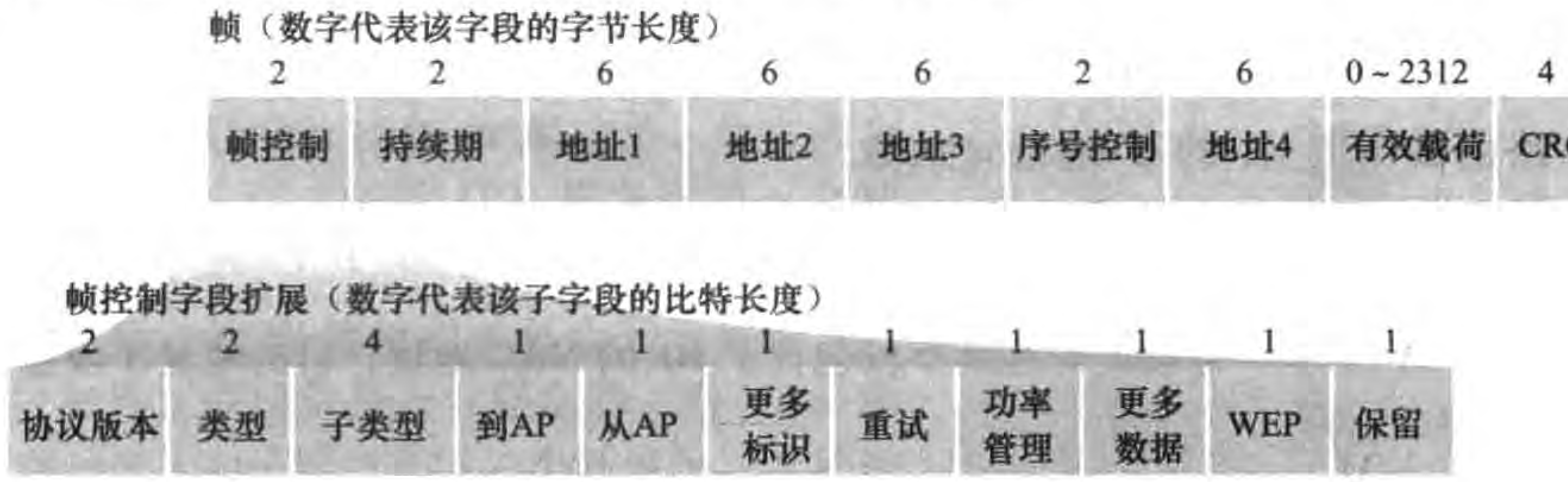


图 7-13 802.11 帧

1. 有效载荷与 CRC 字段

帧的核心是有效载荷，它通常是由一个 IP 数据报或者 ARP 分组组成。尽管这一字段允许的最大长度为 2312 字节，但它通常小于 1500 字节，放置一个 IP 数据报或一个 ARP 分组。如同以太网帧一样，802.11 帧包括一个循环冗余校验（CRC），从而接收方可以检测所收到帧中的比特错误。如我们所看到的那样，比特错误在无线局域网中比在有线局域网中更加普遍，因此 CRC 在这里更加有用。

2. 地址字段

也许 802.11 帧中最引人注意的不同之处是它具有 4 个地址字段，其中每个都可以包含一个 6 字节的 MAC 地址。但为什么要 4 个地址字段呢？如以太网中那样，一个源 MAC 地址字段和一个目的 MAC 地址字段不就足够了？事实表明，出于互联目的需要 3 个地址字段，特别是将网络层数据报从一个无线站点通过一个 AP 送到一台路由器接口。当 AP 在自组织模式中互相转发时使用第四个地址。由于我们这里仅仅考虑基础设施网络，所以只关注前 3 个地址字段。802.11 标准定义这些字段如下：

- 地址 2 是传输该帧的站点的 MAC 地址。因此，如果一个无线站点传输该帧，该站点的 MAC 地址就被插入在地址 2 字段中。类似地，如果一个 AP 传输该帧，该 AP 的 MAC 地址也被插入在地址 2 字段中。
- 地址 1 是要接收该帧的无线站点的 MAC 地址。因此，如果一个移动无线站点传输该帧，地址 1 包含了该目的 AP 的 MAC 地址。类似地，如果一个 AP 传输该帧，地址 1 包含该目的无线站点的 MAC 地址。
- 为了理解地址 3，回想 BSS（由 AP 和无线站点组成）是一个子网的一部分，并且这个

子网经一些路由器接口与其他子网相连。地址3包含这个路由器接口的MAC地址。

为了对地址3的目的有更深入的理解，我们观察在图7-14环境中的网络互联的例子。在这幅图中，有两个AP，每个AP负责一些无线站点。每个AP到路由器有一个直接连接，路由器依次又连接到全球因特网。我们应当记住AP是链路层设备，它既不能"说"IP又不理解IP地址。现在考虑将一个数据报从路由器接口R1移到无线站点H1。路由器并不清楚在它和H1之间有一个AP；从路由器的观点来说，H1仅仅是路由器所连接的子网中的一台主机。

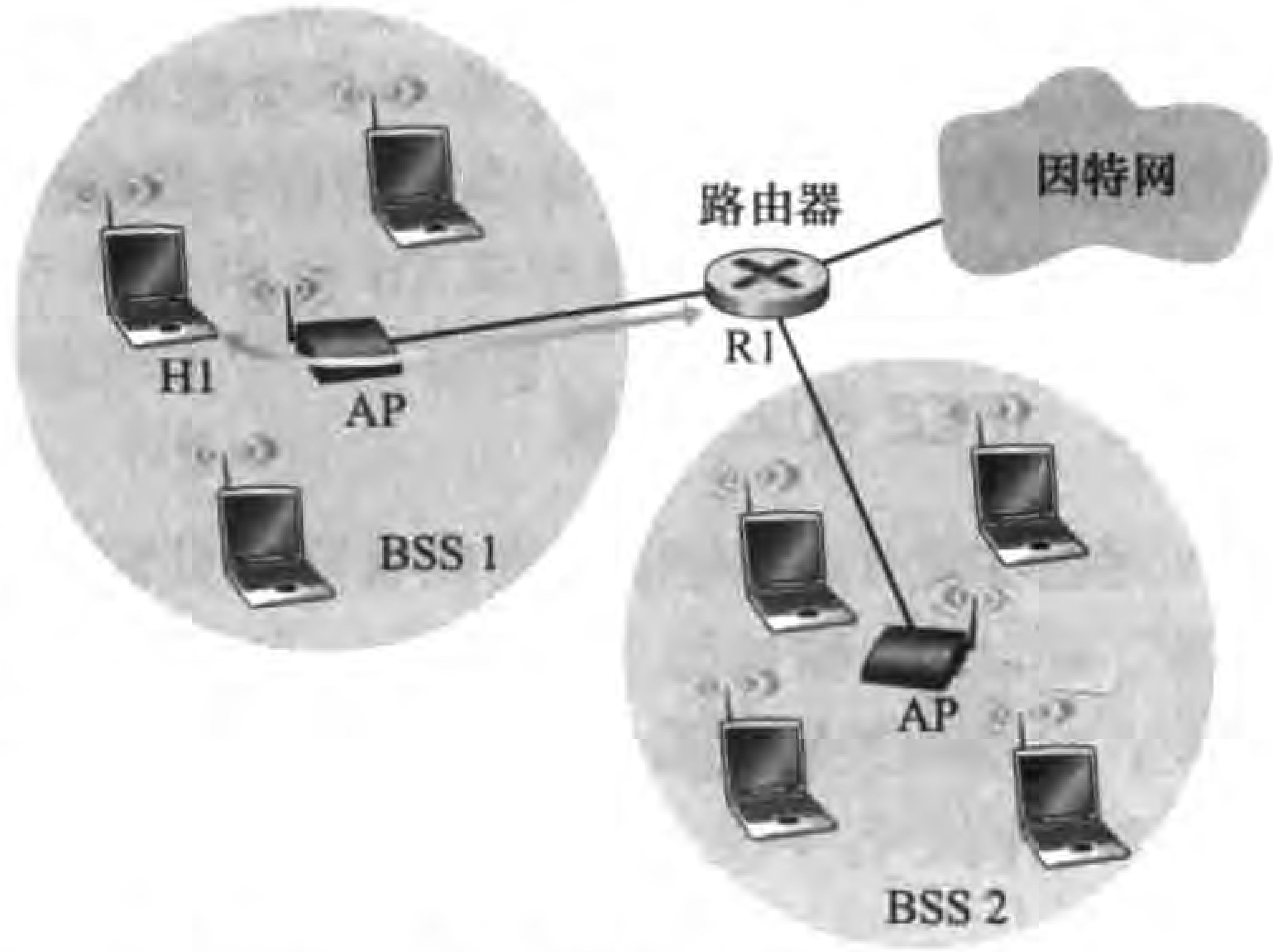


图7-14 在802.11帧中使用地址字段：在H1和R1之间发送帧

- 路由器知道H1的IP地址（从数据报的目的地址中得到），它使用ARP来确定H1的MAC地址，这与在普通的以太网LAN中相同。获取H1的MAC地址后，路由器接口R1将该数据报封装在一个以太网帧中。该帧的源地址字段包含了R1的MAC地址，目的地址字段包含H1的MAC地址。
- 当该以太网帧到达AP后，该AP在将其传输到无线信道前，先将该802.3以太网帧转换为一个802.11帧。如前所述，AP将地址1和地址2分别填上H1的MAC地址和其自身的MAC地址。对于地址3，AP插入R1的MAC地址。通过这种方式，H1可以确定（从地址3）将数据报发送到子网中的路由器接口的MAC地址。

现在考虑在从H1移动一个数据报到R1的过程中无线站点H1进行响应时发生的情况。

- H1生成一个802.11帧，如上所述，分别用AP的MAC地址和H1的MAC地址填充地址1和地址2字段。对于地址3，H1插入R1的MAC地址。
- 当AP接收该802.11帧后，将其转换为以太网帧。该帧的源地址字段是H1的MAC地址，目的地址字段是R1的MAC地址。因此，地址3允许AP在构建以太网帧时能够确定目的MAC地址。

总之，地址3在BSS和有线局域网互联中起着关键作用。

3. 序号、持续期和帧控制字段

前面讲过在802.11网络中，无论何时一个站点正确地收到一个来自于其他站点的帧，它就回发一个确认。因为确认可能会丢失，发送站点可能会发送一个给定帧的多个副本。正如我们在rdt2.1协议讨论中所见（3.4.1节），使用序号可以使接收方区分新传输的帧和以前帧的重传。因此在802.11帧中的序号字段在链路层与在第3章中运输层中的该字段有着完全相同的目的。

前面讲过802.11协议允许传输节点预约信道一段时间，包括传输其数据帧的时间和传输确认的时间。这个持续期值被包括在该帧的持续期字段中（在数据帧和RTS及CTS帧中均存在）。

如图7-13所示，帧控制字段包括许多子字段，我们将提一下其中比较重要的子字段，更加完整的讨论请参见802.11规范［Held 2001；Crow 1997；IEEE 802.11 1999］。类型和

子类型字段用于区分关联、RTS、CTS、ACK 和数据帧。To（到）和 From（从）字段用于定义不同地址字段的含义。（这些含义随着使用自组织模式或者基础设施模式而改变，而且在使用基础设施模式时，也随着是无线站点还是 AP 在发送帧而变化。）最后，WEP 字段指示了是否使用加密（WEP 将在第 8 章中讨论。）

7.3.4 在相同的 IP 子网中的移动性

为了增加无线 LAN 的物理范围，公司或大学经常会在同一个 IP 子网中部署多个 BSS。这自然就引出了在多个 BSS 之间的移动性问题，即无线站点如何在维持进行中的 TCP 会话的情况下，无缝地从一个 BSS 移动到另一个 BSS？正如我们将在本小节中所见，当这些 BSS 属于同一子网时，移动性可以用一种相对直接的方式解决。当站点在不同子网间移动时，就需要更为复杂的移动性管理协议了，我们将在 7.5 节和 7.6 节中学习这些协议。

我们现在看一个同一子网中的不同 BSS 之间的移动性的特定例子。图 7-15 显示了具有一台主机 H1 的两个互联的 BSS，该主机从 BSS1 移动到 BSS2。因为在这个例子中连接两个 BSS 的互联设备不是一台路由器，故在两个 BSS 中的所有站点（包括 AP）都属于同一个 IP 子网。因此，当 H1 从 BSS1 移动到 BSS2 时，它可以保持自己的 IP 地址和所有正在进行的 TCP 连接。如果互联设备是一台路由器，则 H1 必须在它移动进入的子网中获得一个新地址。这种地址的变化将打断（并且最终终止）在 H1 的任何进行中的 TCP 连接。在 7.6 节中，我们将能看到一种网络层移动性协议如移动 IP 能被用于避免该问题。

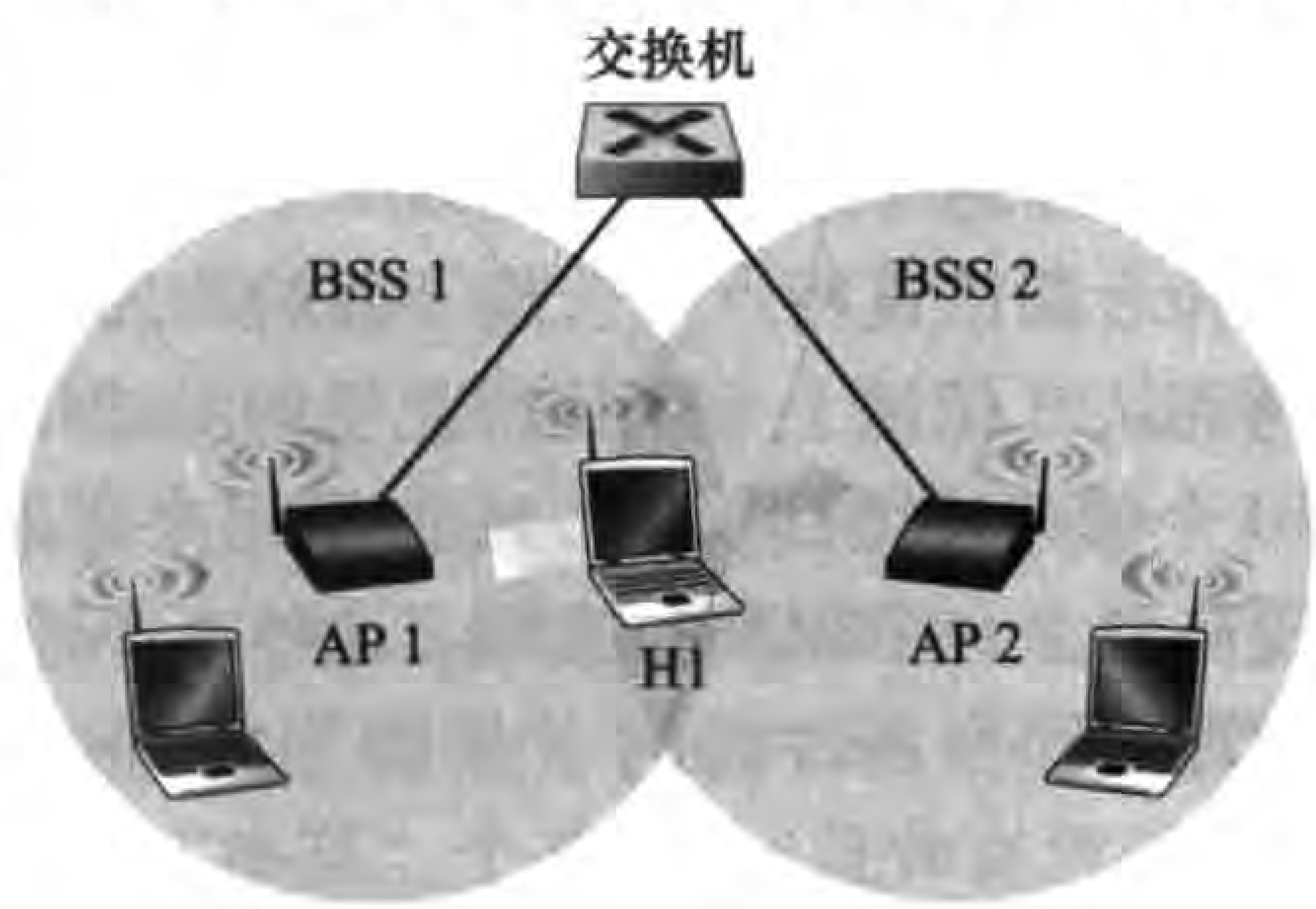


图 7-15　同一子网中的移动性

但是 H1 从 BSS1 移动到 BSS2 时具体会发生哪些事呢？随着 H1 逐步远离 AP1，H1 检测到来自 AP1 的信号逐渐减弱并开始扫描一个更强的信号。H1 收到来自 AP2 的信标帧（在许多公司和大学的设置中它与 AP1 有相同的 SSID）。H1 然后与 AP1 解除关联，并与 AP2 关联起来，同时保持其 IP 地址和维持正在进行的 TCP 会话。

从主机和 AP 的角度，这就处理了切换问题。但对图 7-15 中的交换机又会发生什么样的情况呢？交换机如何知道主机已经从一个 AP 移动到另一个 AP 呢？回想第 6 章所述，交换机是"自学习"的，并且自动构建它们的转发表。这种自学习的特征很好地处理了偶尔的移动（例如，一个雇员从一个部门调转到另一个部门）。然而，交换机没有被设计用来支持用户在不同 BSS 间高度移动，同时又希望保持 TCP 连接。为理解这一问题，回想在移动之前，交换机在其转发表中有一个表项，对应 H1 的 MAC 地址与到达 H1 所通过的出交换机端口。如果 H1 初始在 BSS1 中，则发往 H1 的数据报将经 AP1 导向 H1。然而，一旦 H1 与 BSS2 关联，它的帧应当被导向 AP2。一种解决方法（真有点不规范）是在新的关联形成后，让 AP2 以 H1 的源地址向交换机发送一以太网广播帧。当交换机收到该帧后，更新其转发表，使得 H1 可以通过 AP2 到达。802.11f 标准小组正在开发一个 AP 间的协议来处理这些以及相关的问题。

我们以上的讨论关注了在相同 LAN 子网中的移动性。前面我们在 6.4.4 节学习过 VLAN，它能够用来将若干 LAN 孤岛连接成为一个大虚拟 LAN，该虚拟 LAN 能够跨越很大的地理范围。在这种 VALN 中的基站之间的移动性能够以上述完全相同的方式来处理［Yu 2011］。

7.3.5 802.11 中的高级特色

我们将简要地讨论 802.11 网络中具有的两种高级能力，以此来完成我们学习 802.11 的内容。如我们所见，这些能力并不是完全特定于 802.11 标准的，而是在该标准中可能由特定机制产生的。这使得不同的厂商可使用他们自己（专用）的方法来实现这些能力，这也许能让他们增强竞争能力。

1. 802.11 速率适应

我们在前面图 7-3 中看到，不同的调制技术（提供了不同的传输速率）适合于不同的 SNR 情况。考虑这样一个例子，一个 802.11 用户最初离基站 20 米远，这里信噪比高。在此高信噪比的情况下，该用户能够与基站使用可提供高传输速率的物理层调制技术进行通信，同时维持低 BER。这个用户多么幸福啊！假定该用户开始移动，向离开基站的方向走去，随着与基站距离的增加，SNR 一直在下降。在这种情况下，如果在用户和基站之间运行的 802.11 协议所使用的调制技术没有改变的话，随着 SNR 减小，BER 将高得不可接受，最终，传输的帧将不能正确收到。

由于这个原因，某些 802.11 实现具有一种速率自适应能力，该能力自适应地根据当前和近期信道特点来选择下面的物理层调制技术。如果一个节点连续发送两个帧而没有收到确认（信道上一个比特差错的隐式指示），该传输速率降低到前一个较低的速率。如果 10 个帧连续得到确认，或如果用来跟踪自上次降速以来时间的定时器超时，该传输速率提高到上一个较高的速率。这种速率适应机制与 TCP 的拥塞控制机制具有相同的“探测”原理，即当条件好时（反映为收到 ACK），增加传输速率，除非某个“坏事”发生了（ACK 没有收到）；当某个“坏事”发生了，减小传输速率。因此，802.11 的速率适应和 TCP 的拥塞控制类似于年幼的孩子，他们不断地向父母要求越来越多（如幼儿要糖果，青少年要求推迟睡觉），直到父母亲最后说“够了！”，孩子们不再要求了（仅当以后情况已经变好了才会再次尝试）。已经提出了一些其他方案以改善这个基本的自动速率调整方案 [Kamerman 1997; Holland 2001; Lacage 2004]。

2. 功率管理

功率是移动设备的宝贵资源，因此 802.11 标准提供了功率管理能力，以使 802.11 节点的侦听、传输和接收功能以及其他需要“打开”电路的时间量最小化。802.11 功率管理按下列方式运行。一个节点能够明显地在睡眠和唤醒状态之间交替（像在课堂上睡觉的学生！）。通过将 802.11 帧首部的功率管理比特设置为 1，某节点向接入点指示它将打算睡眠。设置节点中的一个定时器，使得正好在 AP 计划发送它的信标帧前唤醒节点（前面讲过 AP 通常每 100ms 发送一个信标帧）。因为 AP 从设置的功率传输比特知道哪个节点打算睡眠，所以该 AP 知道它不应当向这个节点发送任何帧，先缓存目的地为睡眠主机的任何帧，待以后再传输。

在 AP 发送信标帧前，恰好唤醒节点，并迅速进入全面活动状态（与睡觉的学生不同，这种唤醒仅需要 250μs [Kamerman 1997]）。由 AP 发送的信标帧包含了帧被缓存在 AP 中的节点的列表。如果某节点没有缓存的帧，它能够返回睡眠状态。否则，该节点能够通过向 AP 发送一个探询报文明确地请求发送缓存的帧。对于信标之间的 100ms 时间来说，250μs 的唤醒时间以及类似的接收信标帧及检查以确保不存在缓存帧的短小时间，没有帧要发送和接收的节点能够睡眠 99% 的时间，从而大大节省了能源。

7.3.6 个人域网络：蓝牙和 ZigBee

如图 7-2 所示，IEEE 802.11 WiFi 标准主要针对相距多达 100m 的设备间的通信（当使用 802.11 具有定向天线的点对点配置时除外）。两个其他的 IEEE 802 无线协议是蓝牙和 ZigBee（定义在 IEEE 802.15.1 和 IEEE 802.15.4 标准中［IEEE 802.15 2012］）。

1. 蓝牙

IEEE 802.15.1 网络以低功率和低成本在小范围内运行。它本质上是一个低功率、小范围、低速率的“电缆替代”技术，用于计算机与其无线键盘、鼠标或其他外部设备如蜂窝电话、扬声器、头戴式耳机及其他设备的互联，而 802.11 是一个较高功率、中等范围、较高速率的“接入”技术。为此，802.15.1 网络有时被称为**无线个人域网络**（Wireless Personal Area Network，WPAN）。802.15.1 的链路层和物理层基于早期用于个人域网络的**蓝牙**（Bluetooth）规范［Held 2001，Bisdikian 2001］。802.15.1 网络以 TDM 方式工作于无须许可证的 2.4GHz 无线电波段，每个时隙长度为 625μs。在每个时隙内，发送方利用 79 个信道中的一个进行传输，同时从时隙到时隙以一个已知的伪随机方式变更信道。这种被称作**跳频扩展频谱**（Frequency-Hopping Spread Spectrum，FHSS）的信道跳动的形式将传输及时扩展到整个频谱。802.15.1 能够提供高达 4Mbps 的数据率。

802.15.1 网络是自组织网络：不需要网络基础设施（如一个接入点）来互连 802.15.1 设备。因此，802.15.1 设备必须自己进行组织。802.15.1 设备首先组织成一个多达 8 个活动设备的**皮可网**（piconet），如图 7-16 所示。这些设备之一被指定为主设备，其余充当从设备。主节点真正控制皮可网，即它的时钟确定了皮可网中的时间，它可以在每个奇数时隙中发送，而从设备仅当主设备在前一时隙与其通信后才可以发送，并且只能发送给主设备。除了从设备，网络中还可以有多达 255 个的寄放（parked）设备。这些设备仅当其状态被主节点从寄放转换为活动之后才可以进行通信。

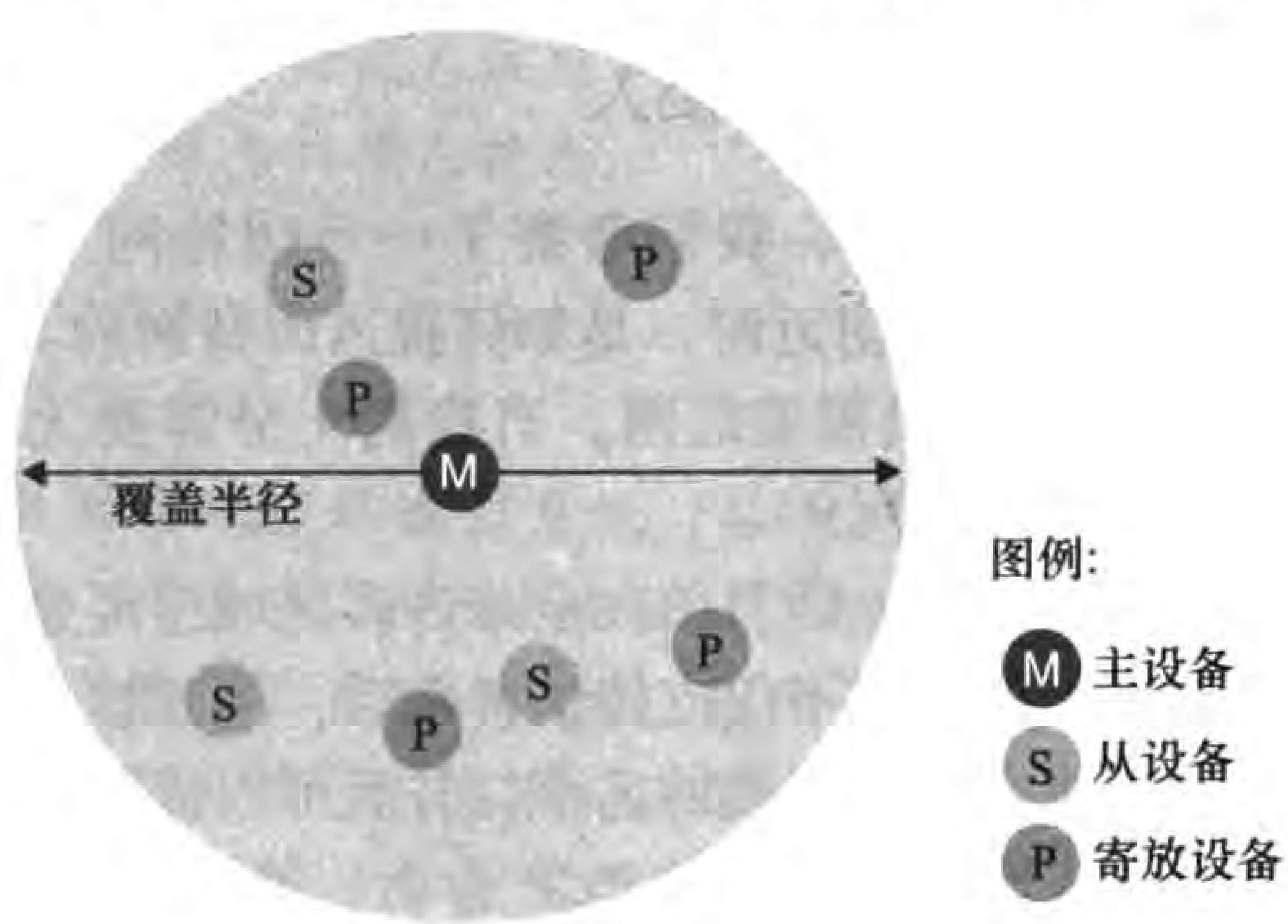


图 7-16　蓝牙皮可网

希望了解更多有关 802.15.1WPAN 信息的读者可以查阅蓝牙参考资料［Held 2001，Bisdikian 2001］，或者 IEEE 802.15 官方 Web 网站［IEEE 802.15 2012］。

2. ZigBee

IEEE 标准化的第二个个人域网络是 802.14.5［IEEE 802.15 2012］，它被称为 ZigBee。虽然蓝牙网络提供了一种“电缆替代”的超过每秒兆比特的数据率，但 ZigBee 较之蓝牙其服务目标是低功率、低数据率、低工作周期的应用。尽管我们可能倾向于认为“更大和更快就更好”，但是并非所有的网络应用都需要高带宽和随之而来的高成本（经济和功率成本）。例如，家庭温度和光线传感器、安全设备和墙上安装的开关都是非常简单、低功率、低工作周期、低成本设备。ZigBee 因此是非常适合于这些设备的。ZigBee 定义了 20kbps、40kbps、100kbps 和 250kbps 的信道速率，这取决于信道的频率。

ZigBee 网络中的节点具有两个特色。多个所谓“简化功能设备”在单个“全功能设

备”控制下作为从设备运行，与蓝牙从设备非常相似。一个全功能设备能够作为一个主设备运行，就像在蓝牙中控制多个从设备那样，并且多个全功能设备还能够配置为一个网状（mesh）网络，其中全功能设备在它们之间发送帧。ZigBee 可以共享许多我们已经在其他链路层协议中遇到的协议机制：信标帧和链路层确认（类似于 802.11），具有二进制回退的载波侦听随机访问协议（类似于 802.11 和以太网），以及时隙的固定、确保的分配（类似于 DOCSIS）。

ZigBee 网络能够配置为许多不同的方式。我们考虑一种简单的场合，其中单一的全功能设备使用信标帧以一种时隙方式控制多个简化功能设备。图 7-17 显示了这种情况，其中 ZigBee 网络将时间划分为反复出现的超帧，每个超帧以一个信标帧开始。每个信标帧将超帧划分为一个活跃周期（在这个周期内设备可以传输）和一个非活跃周期（在这个周期内所有设备包括控制器能够睡眠进而保存能量）。活跃周期由 16 个时隙组成，其中一些由采用 CSMA/CA 随机接入方式的设备使用，其中一些由控制器分配给特定的设备，因而为那些设备提供了确保的信道。有关 ZigBee 网络的更多细节能够在［Baronti 2007，IEEE 802.15.4 2012］中找到。

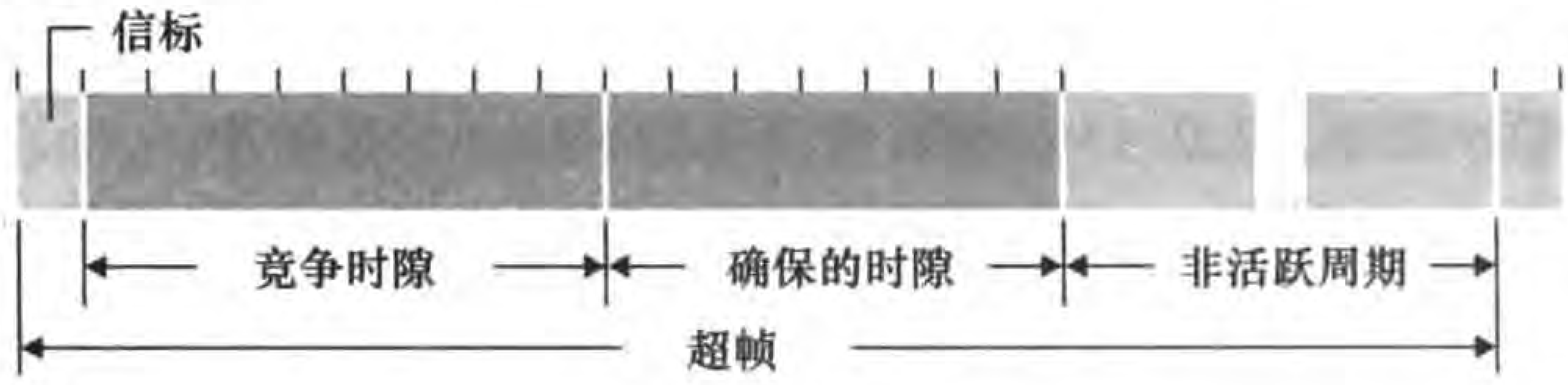


图 7-17　ZigBee 802.14.4 超帧结构

7.4　蜂窝因特网接入

在前一节中，我们考察了一台因特网主机当位于 WiFi 热区中时，即当它位于一个 802.11 接入点附近时，是如何接入因特网的。然而大多数 WiFi 热区只有一个直径为 10～100m 的小规模覆盖范围。当我们十分需要无线因特网接入但同时又无法访问 WiFi 热区时，该怎么办呢？

鉴于蜂窝电话目前在全球许多区域已经是无处不在了，一个很自然的策略就是扩展蜂窝网络，使它们不仅支持语音电话，同时也支持无线因特网接入。理想情况下，这种因特网接入将会有相当高的速率，并且可以提供无缝的移动性，允许用户在旅行过程中（如在汽车或火车上）保持其 TCP 会话。使用足够高的上行和下行比特速率，用户甚至可以在移动中维持视频会议。这种情况并非遥不可及。随着那些我们将在这里涉及的宽带数据服务的更广泛部署，每秒几兆比特的数据速率正变得可行。

在本节中，我们对当前和即将出现的蜂窝因特网接入技术进行简要概述。我们这里仍然重点关注无线第一跳以及将无线第一跳连接进更大的电话网和因特网的网络；在 7.7 节中，我们将考虑如何把呼叫路由选择到在不同基站间移动的用户。我们的简要讨论只是对蜂窝技术进行一个简单、宏观描述。当然，现代蜂窝通信有更大的广度和深度，有许多大学提供关于这一主题的许多课程。希望对此做更深入了解的读者可参阅［Goodman 1997；Kaaranen 2001；Lin 2001；Korhonen 2003；Schiller 2003；Palat 2009；Scourias 2012；Turner 2012；Akyildiz 2010］，以及特别优秀和详尽的参考资料［Mouly 1992；Sauter 2014］。

历史事件

4G 蜂窝移动与无线 LAN 的比较

许多蜂窝移动电话的运营者正在部署4G蜂窝移动系统。在某些国家（如韩国和日本），4G LTE 覆盖率高于90%，即几乎是无所不在。在2015年，已部署的LTE系统的平均下载速率，范围从美国和印度的10Mbps到新西兰的接近40Mbps。这些4G系统部署在需要许可证的无线频带中，运营者向政府支付可观的费用来获得使用频谱的许可证。4G系统以一种与现在仅蜂窝电话接入的相似方式，允许用户在移动中从遥远的户外位置接入因特网。在许多场合中，用户可以同时接入无线 LAN 和 4G。对于4G系统具有的更为受限和更为昂贵的能力，许多移动设备在两者都可用时，默认使用 WiFi 而不是使用4G。无线边缘网络接入将主要经过无线 LAN 还是经过蜂窝系统仍是一个悬而未决的问题：

- 新兴的无线 LAN 基础设施将可能变得几乎无所不在。工作于54Mbps和更高速率的 IEEE 802.11 无线 LAN 已经得到了广泛部署。几乎所有便携计算机、平板电脑和智能手机出厂时都配有802.11 LAN 的能力。而且，新兴的因特网装置（例如无线照相机和相框）也具有低功率的无线 LAN 能力。
- 无线 LAN 的基站也能处理移动电话装置。许多电话已经能够直接或使用类 Skype IP 语音服务与蜂窝电话网络或 IP 网络连接，因此绕过运营者的蜂窝语音和4G数据服务。

当然，许多其他的专家相信4G不仅将取得巨大的成功，而且也将使我们工作和生活的方式发生引人注目的革命。WiFi 和 4G 很可能都会成为流行的无线技术，让漫游无线设备自动选择在其当前所处物理位置提供最好服务的接入技术。

7.4.1 蜂窝网体系结构概述

在本节描述蜂窝网体系结构时，我们将采用全球移动通信系统（GSM）标准的术语。从历史的角度看，首字母缩写词 GSM 源于术语“Groupe Spécial Mobile”，后来才采用了更为英文化的名字，使最初的首字母缩写词得以保留。到了20世纪80年代，欧洲人认识到需要一个泛欧洲的数字蜂窝电话系统，以代替多个不兼容的模拟蜂窝电话系统，从而导致了 GSM 标准的出现［Mouly 1992］。欧洲人在20世纪90年代初就成功地部署了 GSM 技术，自此后 GSM 成长为移动电话领域的庞然大物，全世界有超过80%的蜂窝用户使用 GSM。

当人们谈论蜂窝技术时，经常将该技术分类为几“代”之一。最早一代的设计主要用于语音通信。第一代（1G）系统是模拟 FDMA 系统，其专门用于语音通信。这些1G系统目前几乎绝迹，它们被数字2G系统所替代。初始的2G系统也是为语音而设计，但后来除了语音服务外还扩展了对数据（即因特网）的支持（2.5G）。3G系统也支持语音和数据，但更为强调数据能力和更高速的无线电接入链路。今天正在部署的4G系统基于 LTE 技术，其特征为全 IP 核心网络，并且以几兆比特速率提供了话音和数据集成。

2G 蜂窝网体系结构：语音与电话网连接

术语蜂窝（cellular）是指这样的事实，即由一个蜂窝网覆盖的区域被分成许多称作小

区（cell）的地理覆盖区域，小区如图7-18左侧的六边形所示。如同在7.3.1节中学习的802.11WiFi标准一样，GSM有自己的特殊命名法。每个小区包含一个**收发基站**（Base Transceiver Station，BTS），负责向位于其小区内的移动站点发送或接收信号。一个小区的覆盖区域取决于许多因素，包括BTS的发射功率、用户设备的传输功率、小区中的障碍建筑物以及基站天线的高度。尽管图7-18中显示的是每个小区包含一个位于该小区中间的收发基站，但今天的许多系统将BTS放置在3个小区的交叉处，使得具有有向天线的单个BTS能够为三个小区提供服务。

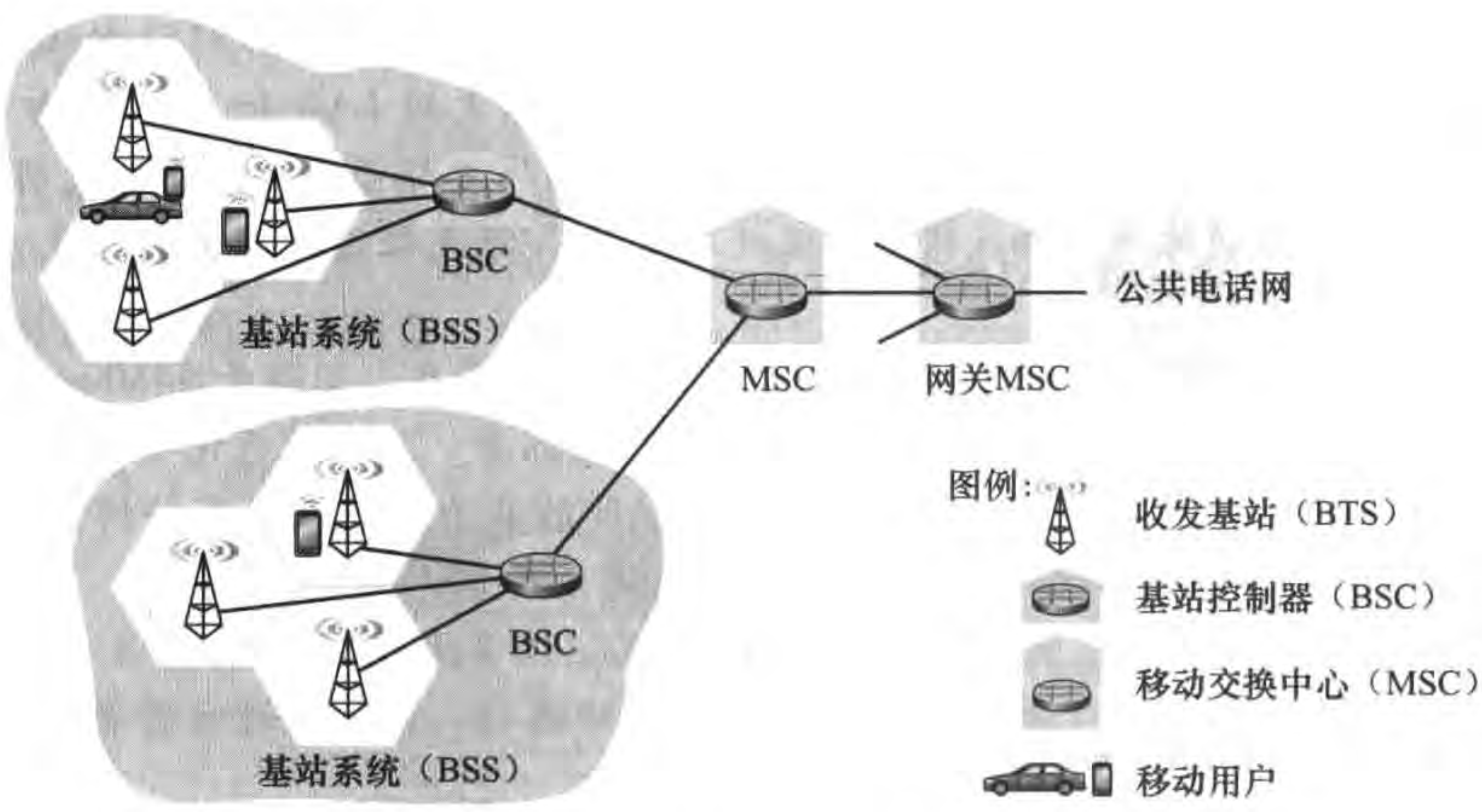


图7-18　GSM 2G蜂窝网体系结构的组件

2G蜂窝系统的GSM标准对空中接口使用了组合的FDM/TDM（无线电）。第1章中讲过，使用纯FDM，信道被划分成许多频段，每个呼叫分配一个频段。第1章也讲过，使用纯TDM，时间被划分为帧，每个帧又被进一步划分为时隙，每个呼叫在循环的帧中被分配使用特定的时隙。在组合的FDM/TDM系统中，信道被划分为若干频率子带；对于每个子带，时间又被划分为帧和时隙。因此，对于一个组合的FDM/TDM系统，如果信道被划分为F个子带，并且时间被划分为T个时隙，那么该信道将能够支持$F \cdot T$个并发的呼叫。我们在6.3.4节中看到，电缆接入网也使用了组合的FDM/TDM方法。GSM系统由多个200kHz的频带组成，每个频带支持8个TDM呼叫。GSM以13kbps和12.2kbps的速率编码。

一个GSM网络的**基站控制器**（Base Station Controller，BSC）通常服务于几十个收发基站。BSC的责任是为移动用户分配BTS无线信道，执行**寻呼**（paging）（找出某移动用户所在的小区），执行移动用户的切换（切换是我们将在7.7.2节中涉及的主题）。基站控制器及其控制的收发基站共同构成了GSM**基站系统**（Base Station System，BSS）。

我们将在7.7节中看到，在用户鉴别和账户管理（决定是否允许某个移动设备与蜂窝网络连接）以及呼叫建立和切换中，**移动交换中心**（Mobile sWitching Center，MSC）起着决定性的作用。单个MSC通常将包含多达5个BSC，因此每个MSC有大约200 000个用户。一个蜂窝提供商的网络将有若干MSC，使用称为网关MSC的特殊MSC将提供商的蜂窝网络与更大的公共电话网相连。

7.4.2　3G蜂窝数据网：将因特网扩展到蜂窝用户

我们在7.4.1节中的讨论关注了蜂窝语音用户连接到公共电话网。但当我们开始这样

干时，当然也乐意读电子邮件、访问 Web、获取位置相关的服务（例如地图和餐馆推荐），或许甚至观看流式视频。为此，我们的智能手机需要运行完整的 TCP/IP 协议栈（包括物理层、链路层、网络层、运输层和应用层），并能够经过蜂窝数据网连接进入因特网。随着一代（和半代）继承一代，以及引入许多具有新首字母缩略词的新技术和服务，蜂窝数据网的主题是一个相当令人眼花缭乱的竞争和不断演化的标准集合，这个标准集合相当令人迷惑。更糟糕的是，没有单一的官方机构对 2.5G、3G、3.5G 或 4G 技术提出要求，难以理清这些竞争性标准之间的差异。在我们下面的讨论中，我们将关注由第三代合作伙伴项目（3rd Generation Partnership Project，3GPP）研发的通用移动通信服务（Universal Mobile Telecommunications Service，UMTS）3G 和 4G 标准［3GPP 2016］。

我们自上而下地查看一下显示在图 7-19 中的 3G 蜂窝数据网体系结构。

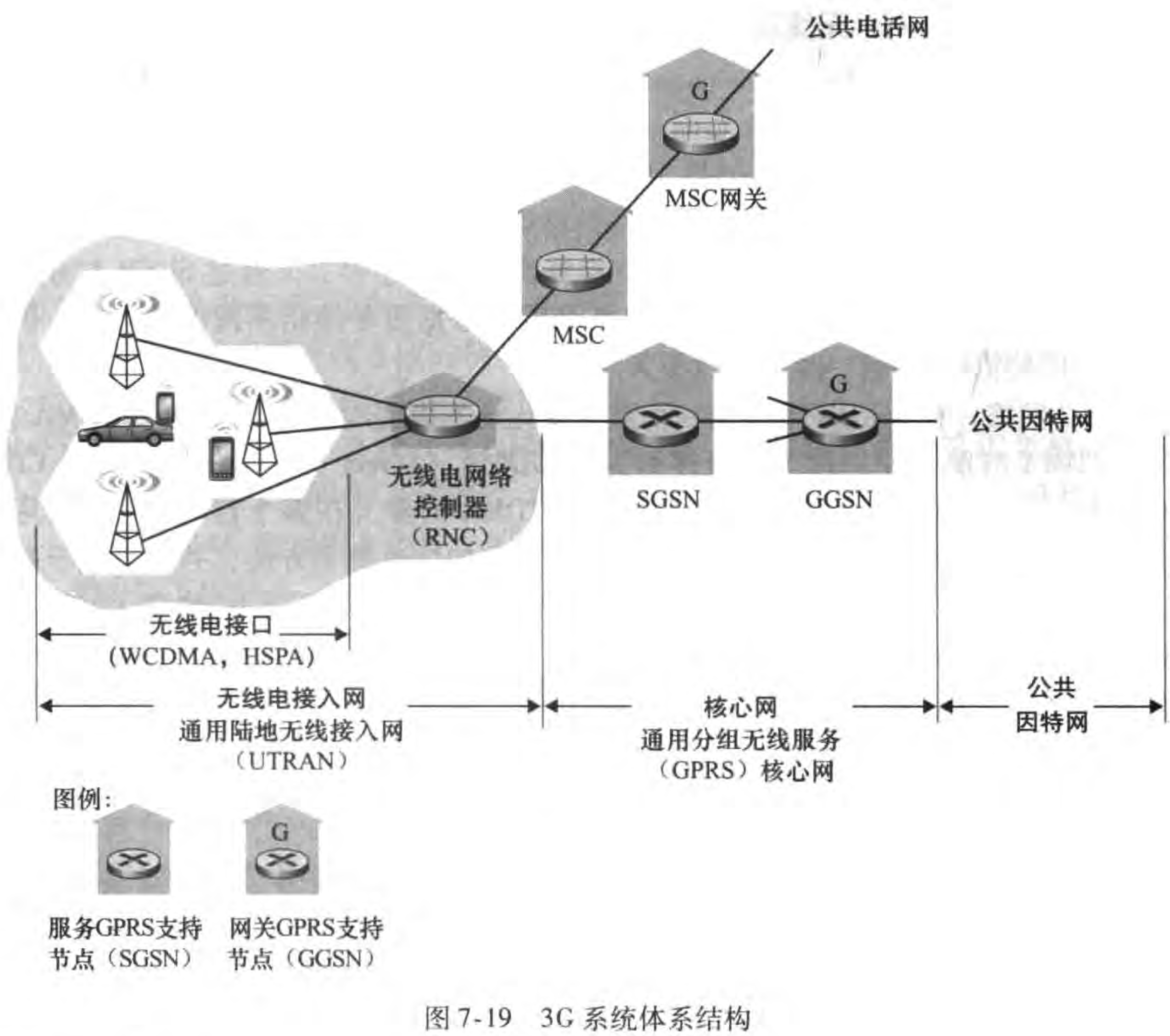


图 7-19 3G 系统体系结构

1. 3G 核心网

3G 核心蜂窝数据网将无线电接入网连接到公共因特网。核心网与我们前面在图 7-18 中遇到过的现有蜂窝语音网（特别是 MSC）的组件协作。由于在现有的蜂窝语音网中具有大量的现有基础设施（有利可图的服务!），3G 数据服务的设计者们所采取的方法非常清楚：*不去触动现有核心 GSM 蜂窝语音网，增加与现有蜂窝语音网平行的附加蜂窝数据功能。*如果将新的数据服务直接增加到现有的蜂窝语音网上，这种方法同样会引发 4.3 节中遇到过的挑战——在前面我们讨论了在因特网中新的（IPv6）和旧的（IPv4）集成的技术。

在3G核心网中有两类节点：**服务通用分组无线服务支持节点**（Serving Generalized packet radio service Support Node，SGSN）和**网关GPRS支持节点**（Gateway GPRS Support Node，GGSN）。（GPRS（General Packet Radio Service）表示通用分组无线服务，这是一种在2G网络中的早期蜂窝数据服务；这里我们讨论的是在3G网络中GPRS的演化版本。）一个SGSN负责向位于其连接的无线电接入网中的移动节点交付（或从移动节点获取）数据报。SGSN与该区域蜂窝语音网的MSC进行交互，提供用户认证和切换，维护活跃移动节点的位置（小区）信息，执行位于无线接入网中的移动节点和GGSN之间的数据报转发。GGSN起着网关的作用，将多个SGSN连接到更大的因特网。GGSN因此是源于移动节点的一个数据报在进入更大因特网之前遇到的3G基础设施的最后一部分。对外部而言，GGSN看起来像任何其他网络路由器，从外面看来，GGSN网络中3G节点的移动性隐藏在GGSN背后。

2. 3G无线电接入网：无线边缘

3G**无线电接入网**（radio access network）是我们作为3G用户看见的无线第一跳网络。**无线电网络控制器**（Radio Network Controller，RNC）通常控制几个小区的收发基站，类似于我们在2G网络中遇到的基站（但是3G UMTS的正式用语称为一个“节点B”，这是一个相当不具描述性名字!）。每个小区的无线链路运行在移动节点和收发基站之间，就像在2G网络中那样。RNC既通过MSC与电路交换蜂窝语音网连接，又通过SGSN与分组交换的因特网连接。因此，尽管3G蜂窝语音服务和蜂窝数据服务使用不同的核心网，但它们共享一个相同的第一/最后一跳无线电接入网。

较之2G网络，在3G UMTS中的一个重大变化是不再使用GSM的FDMA/TDMA方案，UMTS在TDMA时隙中使用称为**直接序列宽带CDMA**（Direct Sequence Wideband CDMA，DS-WCDMA）的CDMA技术［Dahlman 1998］。TDMA时隙又在多个频率上可供使用，即有趣地使用了我们在前面第6章指出的全部三种不同的信道共享方法，并且类似于有线电缆接入网中所采用的方法（参见6.3.4节）。这种变化要求一个新的3G蜂窝无线接入网与显示在图7-19中的2G BSS无线电网络并行运行。与WCDMA规范相关的数据服务被称为**高速分组接入**（High Speed Packet Access，HSPA），其下行数据传输率有望高达14Mbps。有关3G网络的细节能够在3GPP Web站点上找到［3GPP 2016］。

7.4.3 走向4G：LTE

第四代（4G）蜂窝系统正在广泛部署中。在2015年，有50多个国家的4G覆盖率超过50%。由3GPP提出的4G长期演进互联网（LTE）标准［Sauter 2014］，较之3G系统而言有两个重要的创新：一个全IP核心网和一个加强的无线电接入网。下面对此进行讨论。

1. 4G系统体系结构：一个全IP核心网

图7-20显示了总体的4G网络体系结构，它（不幸地）又引入另一个（相当难以理解的）新词汇以及一些网络组件的首字母缩写词。但我们不要被这些首字母缩写词搞晕！有关4G体系结构，有两个重要的高层次观察。

- *一种统一的、全IP网络体系结构*。与图7-19显示的3G网络不同，3G网络对于语音和数据流量具有分离的网络组件和路径，显示在图7-20中的4G体系结构是“全IP的”，即语音和数据都承载在IP数据报中，来自/去往无线设备（用户设

备，4G 的术语为 UE），到分组网关（P-GW）——该 P-GW 将 4G 边缘网络连接到网络的其他部分。使用了 4G，蜂窝网络来源于电话的最后痕迹已经荡然无存，让位给统一的 IP 服务了！

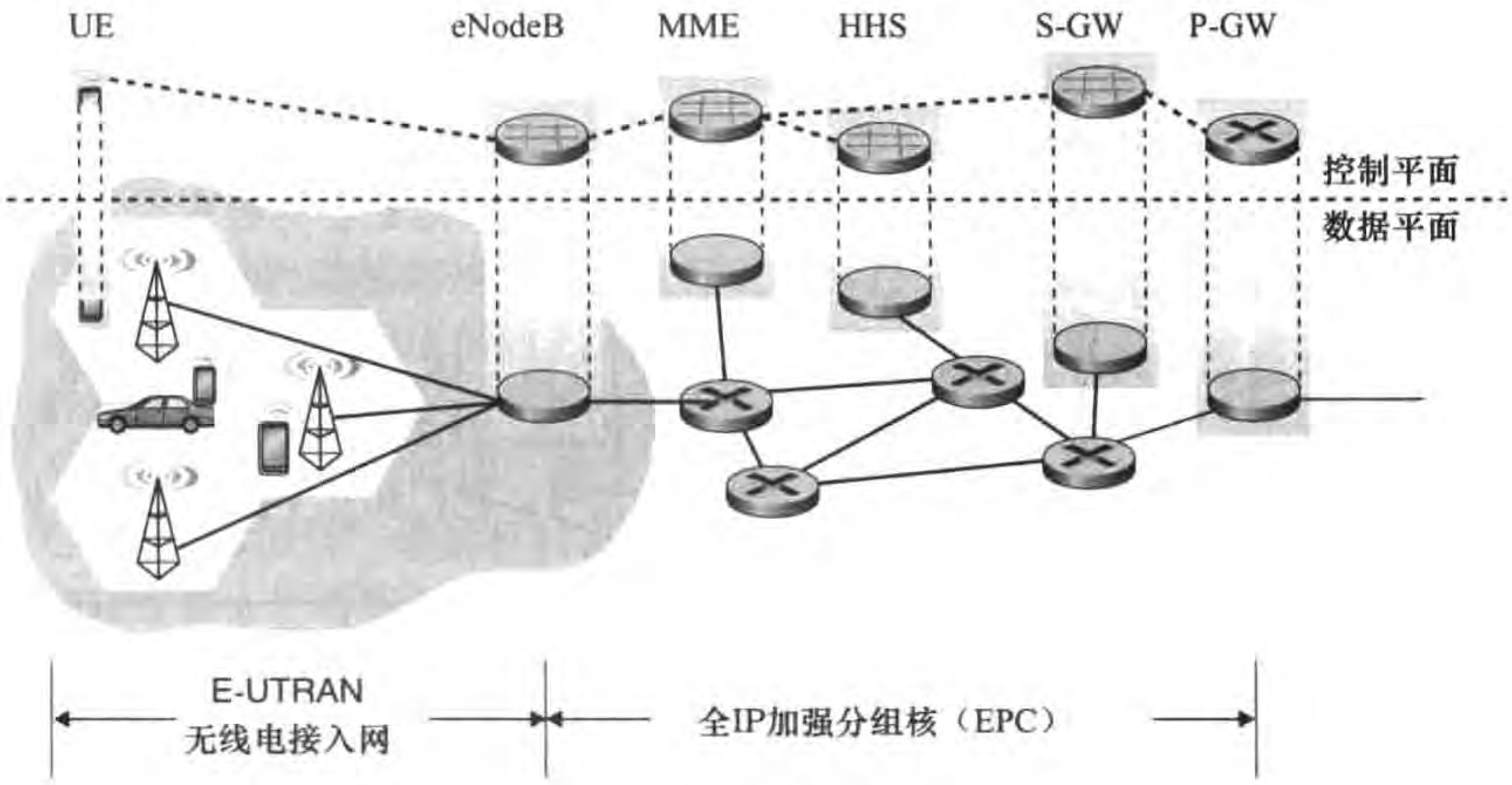


图 7-20 4G 网络体系结构

- *4G 数据平面与 4G 控制平面的清晰分离。*分别对照第 4 章和第 5 章中 IP 网络层的数据平面与控制平面之间的特征，4G 网络体系结构也清晰地分离了数据平面和控制平面。我们将在下面讨论它们的功能。
- *无线电接入网与全 IP 核心网之间的清晰分离。*承载用户数据的 IP 数据报经过通往外部因特网的内部 4G IP 网络，在用户（UE）和网关（图 7-20 中的 P-GW）之间转发。在 4G 控制服务组件之间经过相同的内部网络交换控制分组，这些组件的作用将在下面描述。

4G 体系结构的主要组件如下：

- **eNodeB** 是 2G 基站和 3G 无线电网络控制器（又称为节点 B）的逻辑后代，并且此时还起着关键作用。它的数据平面作用是在 UE 和 P-GW 之间（经过 LTE 无线电接入网）转发数据报。UE 数据报在 eNodeB 被封装，并且通过 4G 网络的全 IP 强化分组核（EPC）以隧道形式传输到 P-GW。eNodeB 与 P-GW 之间的隧道类似于我们在 4.3 节中看到的 IPv6 数据报隧道，这些分组在两个 IPv6 端点之间通过一个使用 IPv4 路由器的网络传输。这些隧道可能与保证服务质量（QoS）相关。例如，4G 网络可能确保语音流量在 UE 和 P-GW 之间历经不超过 100ms 时延，分组丢失率小于 1%；TCP 流量也许能够确保 300ms 时延以及小于 0.0001% 的分组丢失率 [Palat 2009]。我们将在第 9 章涉及 QoS。在控制平面中，eNodeB 代表 UE 来处理注册和移动性信令流量。
- **分组数据网络网关**（Packet Data Network Gateway，P-GW）给 UE 分配 IP 地址，并且保证 QoS 实施。作为隧道端点，当向或从 UE 转发数据报时，它也执行数据报封装/解封装。
- **服务网关**（S-GW）是数据平面移动性锚点，即所有 UE 流量将通过 S-GW 传递。该 S-GW 也执行收费/记账功能以及法定的流量拦截。
- **移动性管理实体**（Mobility Management Entity，MME）代表位于它所控制单元中的

UE，执行连接和移动性管理。它从 HHS 接收 UE 订购信息。我们在 7.7 节中详细讨论蜂窝网络中的移动性。

- **归属用户服务**（Home Subscriber Server，HSS）包含了包括漫游接入能力、服务质量配置文件和鉴别信息的 UE 信息。如我们在 7.7 节中所见，该 HSS 从 UE 归属蜂窝提供商得到这些信息。

[Motorola 2007；Palat 2009；Sauter 2014] 对 4G 网络体系结构及其 EPC 做了可读性很强的介绍。

2. LTE 无线电接入网

LTE 在下行信道采用频分复用和时分复用结合的方法，称之为**正交频分复用**（Orthogonal Frequency Division Multiplexing，OFDM）技术［Rohde 2008；Ericsson 2011］。（"正交"一词来源于如下事实，所生成的在不同频道上发送的信号，它们相互干扰非常小，即使当信道频率紧密排列时）。在 LTE 中，每个活跃的移动节点都可以在一个或更多个信道频率上被分配一个或更多个 0.5ms 时隙。图 7-21 显示了在 4 个频率上分配 8 个时隙的情况。通过分配越来越多的时隙（无论是用相同的频率还是用不同的频率），移动节点能够获取越来越高的传输速率。在移动节点之间进行时隙（重）分配的频度为每毫秒一次。不同的调制方案也能用于改变传输速率，参见我们前面对图 7-3 的讨论以及 WiFi 网络中调制方案的动态选择。

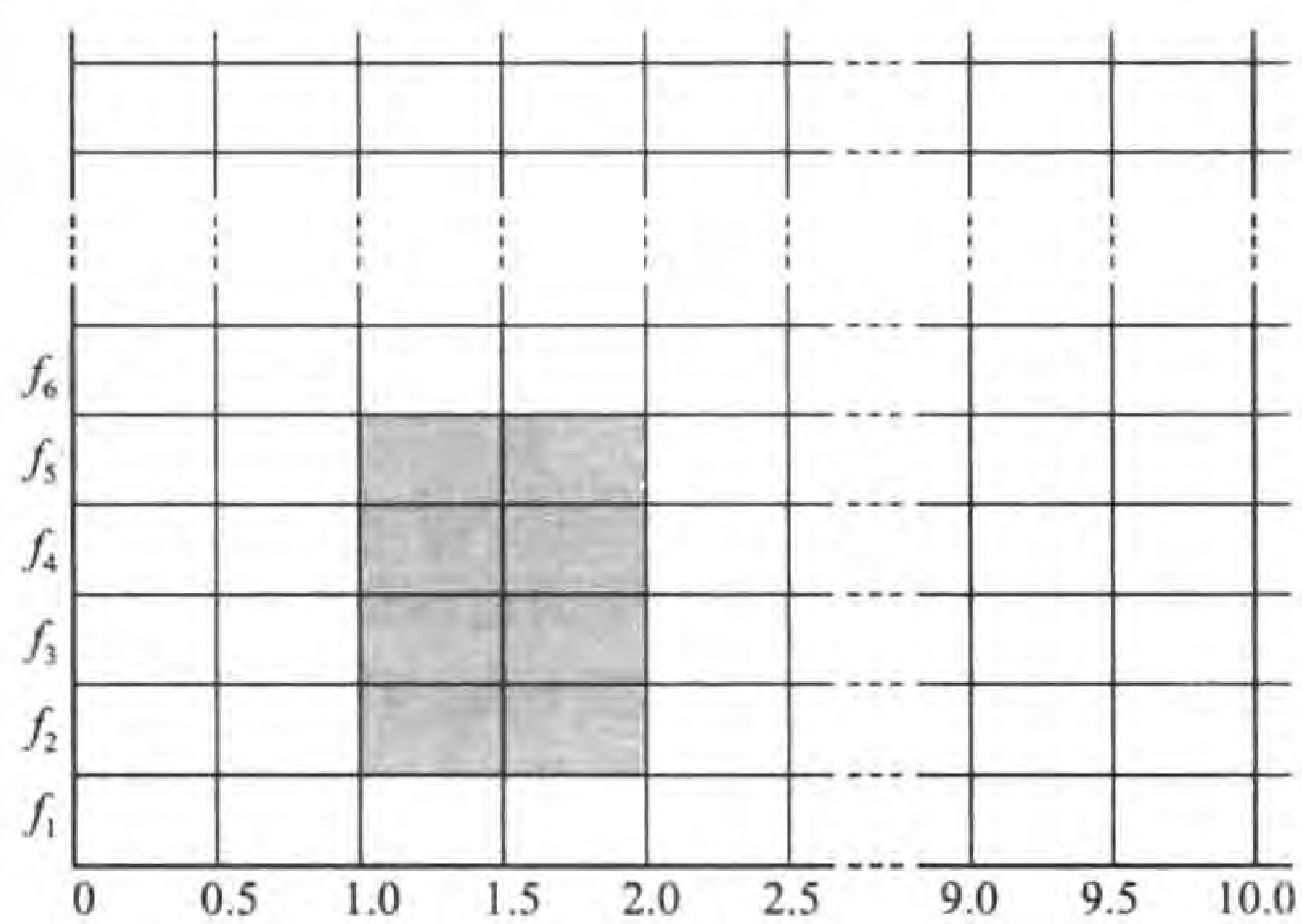


图 7-21 在每个频率上，20 个 0.5ms 的时隙组成 10ms 帧。阴影显示了一个 8 时隙分配

LTE 标准并未对向移动节点特殊分配时隙进行强制要求。相反，允许哪个移动节点在某个给定的时隙在给定的频率下传输，这个决定由 LTE 设备商和/或网络运营商提供的调度算法来做出。使用机会调度［Bender 2000；Kolding 2003；Kulkarni 2005］，将物理层协议与发送方和接收方之间的信道条件相匹配，基于信道条件选择分组将发往的接收方，使无线电网络控制器能够最大限度地利用无线媒体。此外，能够使用用户优先权和服务的契约等级（如银、金或铂金）调度下行分组传输。除了上面描述的 LTE 能力，高级 LTE 通过向移动节点分配聚合信道提供了数百兆下行带宽［Akyildiz 2010］。

另一种 4G 无线技术是 WiMAX（全球微波接入互操作），它是一个 IEEE 802.16 标准协议簇，与 LTE 有着重大差异。WiMAX 目前还没有得到 LTE 那样的广泛部署，它的详细讨论能够在本书的 Web 站点上找到。

7.5 移动管理：原理

学习了无线网络中通信链路的无线特性后，现在我们将注意力转向这些无线链路带来的移动性。宽泛地讲，移动节点是随时间改变它与网络连接位置的节点。因为移动性这一术语在计算机界和电话界有许多含义，所以先更为详细地讨论一下移动性的各个方面将对我们很有帮助。

- *从网络层的角度看，用户如何移动？* 一个物理上移动的用户将对网络层提出一系列不同寻常的挑战，这取决于他（她）在网络连接点之间如何移动。在图 7-22 中的移动程度谱的一端，用户也许带着一台装有无线网络接口卡的便携机在一座建筑物内走动。如我们在 7.3.4 节中所见，从网络层的角度来看，该用户并没有移动。而且，如果该用户不论在何处都与同一个接入点相关联，从链路层角度来看该用户甚至也没有移动。

  在该移动程度谱的另一端，考虑一下该用户在一辆宝马或特斯拉轿车内以 150km/h 的时速沿高速公路急速行驶时穿过多个无线接入网，并希望在整个旅程中保持一个与远程应用的不间断的 TCP 连接。这个用户*无疑是移动的*！在这两种极端之间的情况是，一个用户带着一台便携机从一个地方（如办公室或宿舍）到另一个地方（如咖啡店、教室），并且想在新地方连入网络。该用户也是移动的（虽然比“宝马”驾驶员的移动性差一些！），只不过不需要在网络接入点之间移动时维持一个不间断的连接。图 7-22 从网络层角度阐明了用户移动性的程度谱。



图 7-22　从网络层观点来看各种程度的移动性

- *移动节点的地址始终保持不变有多么重要？* 对移动电话而言，当你从一个提供商移动电话网络到另一个的过程中，你的电话号码（本质上是你电话的网络层地址）始终保持不变。类似地，便携机在 IP 网络之间移动时是否也必须维持相同的 IP 地址呢？

  对这一问题的回答很大程度上取决于所运行的应用程序。对于那个在高速公路上飞驰，同时又希望维持对一个远程应用的不间断的 TCP 连接的宝马或特斯拉司机而言，维持相同的 IP 地址将会带来便利。回想第 3 章，一个因特网应用程序需要知道它与之通信的远端实体的 IP 地址和端口号。如果一个移动实体在移动过程中能够保持其 IP 地址不变，从应用的角度，移动性就变得不可见。这种透明性有十分重要的价值，即应用程序不必关心 IP 地址潜在的变化，并且同样的应用程序代码既可用于移动连接，又可用于非移动连接。在下一节我们将会看到移动 IP 提供了这种透明性，它允许移动节点在网络间移动的同时维持其永久的 IP 地址。

  在另一方面，一个不太喜欢新潮的移动用户也许只想关闭办公室便携机，将其带回家，然后开机，再在家中工作。如果该便携机在家时只是作为一个客户，使用客户 - 服务器方式的应用（如发送/阅读电子邮件、浏览 Web、通过 Telnet 与远程主机相连），则使用特定 IP 地址并不是那么重要。特别是，用户能够得到一个由服务于家庭的 ISP 临时分配的 IP 地址即可。我们在 4.3 节中看到的 DHCP 提供了这种功能。

- *支持有线基础设施的东西有哪些可用？* 在所有上述情形中，我们都隐含地假设存在一个固定的基础设施让移动用户连接，例如家庭的 ISP 网络、办公室的无

线接入网，或者沿高速公路的无线接入网。如果这样的基础设施不存在会怎么样？如果两个用户位于彼此的通信范围内，他们能否在没有其他网络基础设施存在的情况下建立一个网络连接？自组织网络正好提供了这些能力。这一飞速发展的领域位于移动网络研究的前沿，超出了本书的范围。[Perkins 2000] 和 IETF 移动自组织网络（manet）工作组主页 [manet 2016] 提供了有关这一主题的详尽讨论。

为了阐述允许移动用户在不同网络间移动过程中维持正在进行的连接所涉及的问题，我们考虑一个人类的类比例子。一位20岁左右的青年从家里搬出，成为流动的人，在一些宿舍或公寓居住，并经常改换住址。如果一个老朋友想与他联系，这位朋友怎样才能找到这个流动的朋友呢？一种常用的方法是与他的家庭取得联系，因为一位流动的青年通常会将其目前的地址告诉家里（即使没有其他原因，哪怕只是为了让父母寄钱来帮他付房租）。其家庭由于有一个永久地址，因此成为其他想与该流动青年联系的人可采用的第一步。这些朋友后来与他的通信也许是间接的（如先将邮件发送到其父母家，再转发给该流动的青年），也许是直接的（如该朋友用得到的地址直接将邮件发送给其流动的朋友）。

在一个网络环境中，一个移动节点（如一台便携机或智能手机）的永久居所被称为**归属网络**（home network），在归属网络中代表移动节点执行下面讨论的移动管理功能的实体叫**归属代理**（home agent）。移动节点当前所在网络叫作**外部网络**（foreign network）或**被访网络**（visited network），在外部网络中帮助移动节点做移动管理功能的实体称为**外部代理**（foreign agent）。对于移动的专业人员而言，他们的归属网络可能就是其公司网络，而被访网络也许就是他们正访问的某同行所在的网络。一个**通信者**（correspondent）就是希望与该移动节点通信的实体。图7-23阐述了这些概念，也说明了下面考虑的编址概念。在图7-23中，我们注意到代理被配置在路由器上（例如，作为在路由器上运行的进程），但它们也能在网络中其他主机或服务器上执行。

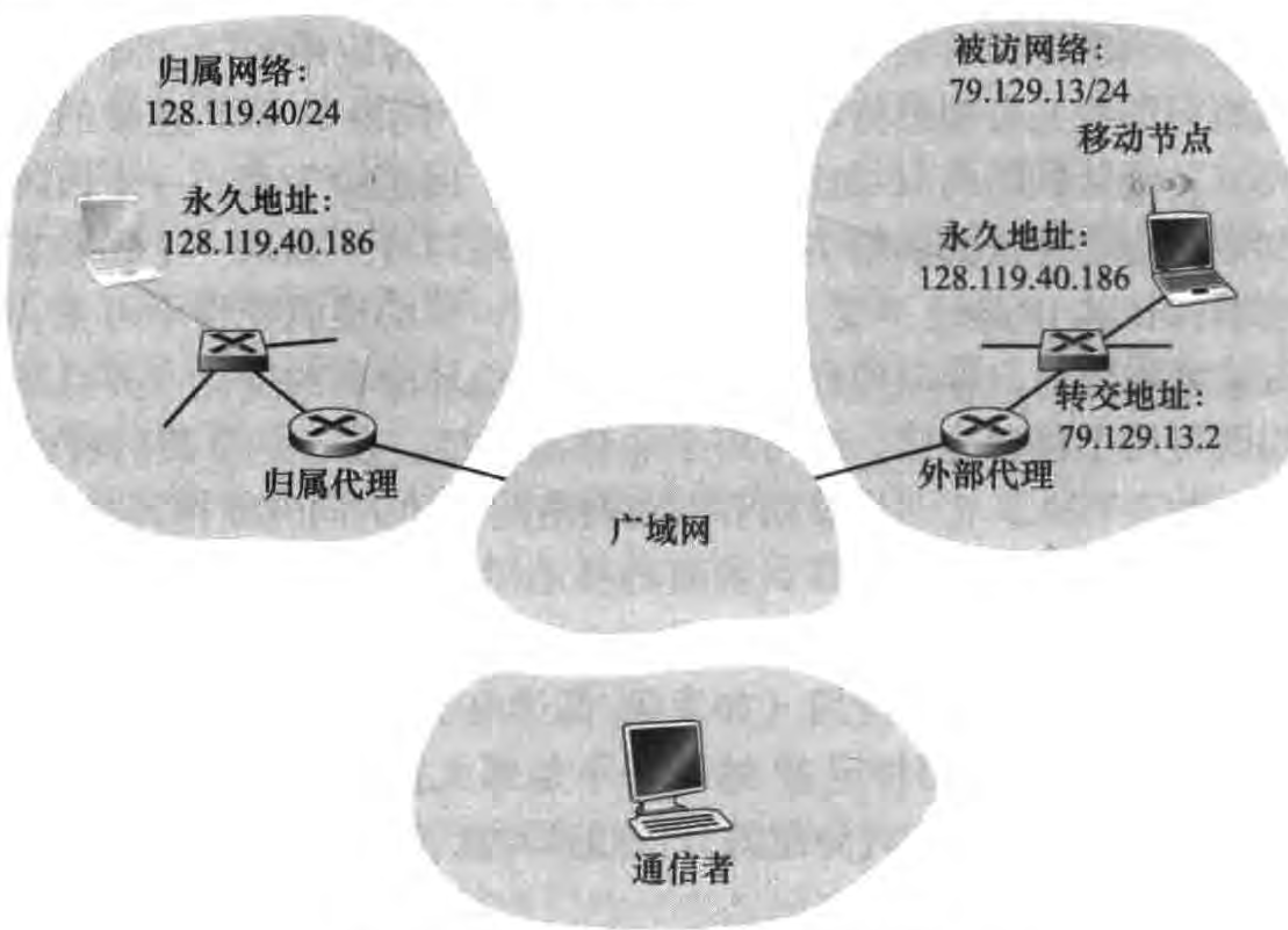


图7-23　移动网络体系结构中的初始要素

7.5.1 寻址

我们前面提到为了使用户移动性对网络应用透明，希望一个移动节点在从一个网络移动到另一个网络时保持其地址不变。当某移动节点位于一个外部网络时，所有指向此节点固定地址的流量需要导向外部网络。怎样才能做到这一点呢？外部网络可用的一种方法就是向所有其他网络发通告，告诉它们该移动节点正在它的网络中。这通常可通过交换域内与域间路由选择信息来实现，而且只需对现有路由选择基础设施做很少的改动即可。外部网络只需通告其邻居它有一条非常特别的路由能到达该移动节点的固定地址，即告诉其他网络它有一条正确的路径可将数据报导向该移动节点的固定地址（即基本上是通知其他网络，它有一条可将数据报路由选择到该移动节点的永久地址的正确路径；参见 4.3 节）。这些邻居将在全网传播该路由选择信息，而且是当作更新路由选择信息和转发表的正常过程的一部分来做。当移动节点离开一个外部网络后又加入另一个外部网络时，新的外部网络会通告一条新的通向该移动节点的特别路由，旧的外部网络将撤销其与该移动节点有关的路由选择信息。

这种方法立刻解决了两个问题，且它这样做不需对网络层基础设施做重大改动。其他网络知道该移动节点的位置，很容易将数据报路由到该移动节点，因为转发表将这些数据报导向外部网络。然而它有一个很大的缺陷，即扩展性不好。如果移动性管理是网络路由器的责任的话，则路由器将必须维护可能多达数百万个移动节点的转发表表项。在本章后面的习题中将探讨一些其他的缺陷。

一种替代的方法（并在实际中得到了采用）是将移动性功能从网络核心搬到网络边缘，这是我们在研究因特网体系结构时一再重复的主题。一种自然的做法是由该移动节点的归属网络来实现。与那个流动青年的父母跟踪他们孩子的位置有许多相似之处，在移动节点的归属网络中的归属代理也能跟踪该移动节点所在的外部网络。这当然需要一个移动节点（或一个代表该移动节点的外部代理）与归属代理之间的协议来更新移动节点的位置。

我们现在更详细地思考外部代理。如图 7-23 所示，概念上最简单的方法是将外部代理放置在外部网络的边缘路由器上。外部代理的作用之一就是为移动节点创建一个所谓的**转交地址**（Care-Of Address，COA），该 COA 的网络部分与外部网络的网络部分相匹配。因此一个移动节点可与两个地址相关联，即其**永久地址**（permanent address）（类比于流动青年的家庭地址）与其 COA，该 COA 有时又称为**外部地址**（foreign address）（类比于流动青年当前居住的房屋地址）。在图 7-23 中的例子中，移动节点的固定地址是 128.119.40.186。当被访网络为 79.129.13/24 时，该移动节点具有的 COA 为 79.129.13.2。外部代理的第二个作用就是告诉归属代理，该移动节点在它的（外部代理的）网络中且具有给定的 COA。我们很快就会看到，该 COA 将用于将数据报通过外部代理“重新路由选择”到移动节点。

虽然我们已将移动节点与外部代理的功能分开，但是应当注意到移动节点也能承担外部代理的责任。例如，某移动节点可在外部网络中得到一个 COA（使用一个诸如 DHCP 之类的协议），且由它自己把其 COA 通告给归属代理。

7.5.2 路由选择到移动节点

我们现在已看到一个移动节点是如何得到一个 COA 的，归属代理又是如何被告知该地址的。但让归属代理知道该 COA 仅能解决部分问题。数据报应怎样寻址并转发给移动

节点呢？因为只有归属代理（而不是全网的路由器）知道该移动节点的位置，故如果只是将一个数据报寻址到移动节点的永久地址并将其发送到网络层基础结构中，这样做已不再满足需要了。还有更多的事情要做。目前有两种不同的方法，我们将称其为间接路由选择与直接路由选择。

1. 移动节点的间接路由选择

我们先考虑一个想给移动节点发送数据报的通信者。在**间接路由选择**（indirect routing）方法中，通信者只是将数据报寻址到移动节点的固定地址，并将数据报发送到网络中去，完全不知道移动节点是在归属网络中还是正在访问某个外部网络，因此移动性对于通信者来说是完全透明的。这些数据报就像平常一样首先导向移动节点的归属网络。这用图7-24中的步骤1加以说明。

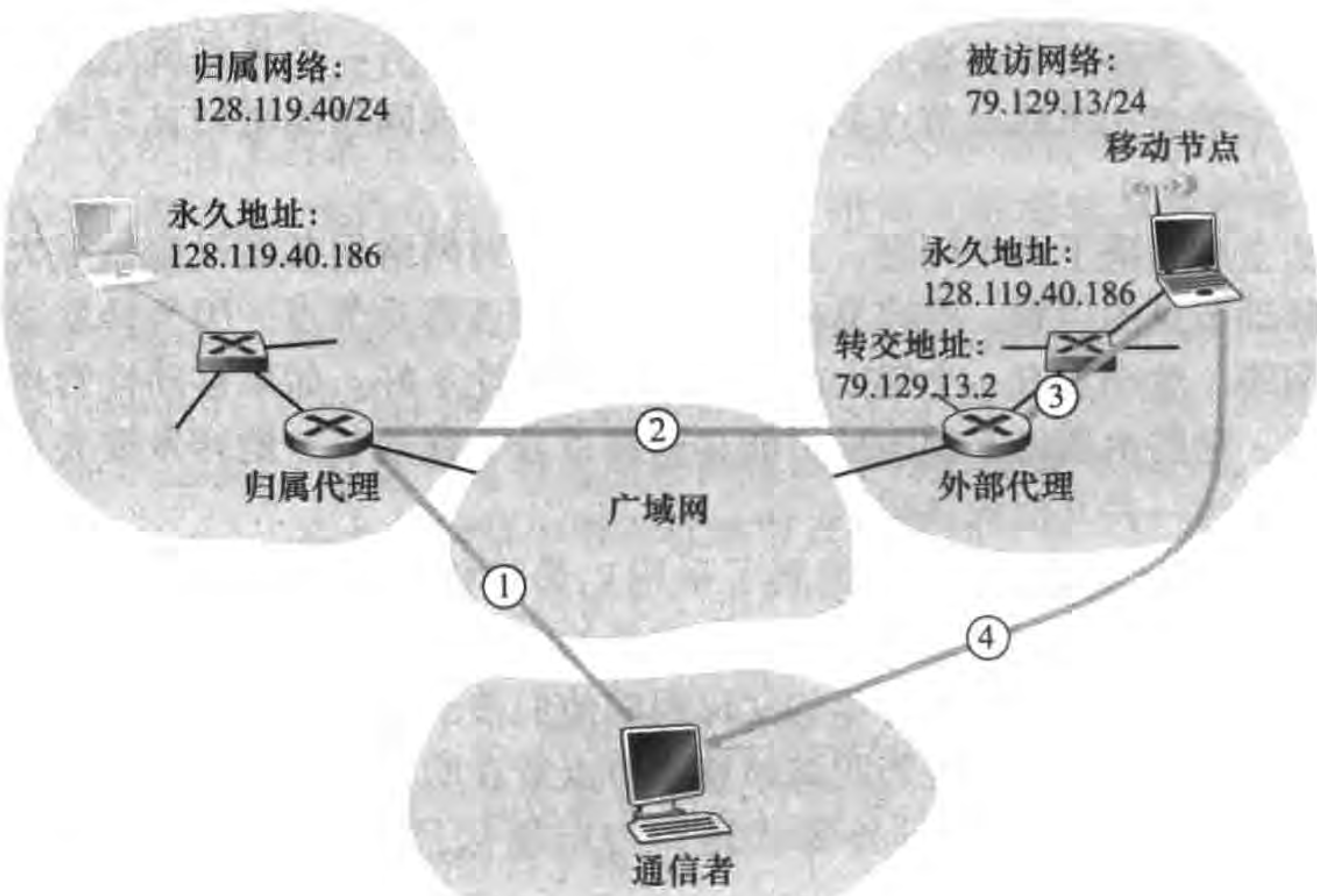


图7-24 对移动节点的间接路由选择

我们现在将注意力转向归属代理。除了负责与外部代理交互以跟踪移动节点的COA外，归属代理还有另一项很重要的功能。它的第二项工作就是监视到达的数据报，这些数据报寻址的节点的归属网络与该归属代理所在网络相同，但这些节点当前却在某个外部网络中。归属代理截获这些数据报，然后按一个两步骤的过程转发它们。通过使用移动节点的COA，该数据报先转发给外部代理（图7-24中的步骤2），然后再从外部代理转发给移动节点（图7-24中的步骤3）。

仔细地思考这种重新路由选择过程是有益的。归属代理需要用该移动节点的COA来设置数据报地址，以便网络层将数据报路由选择到外部网络。在另一方面，需要保持通信者数据报的原样，因为接收该数据报的应用程序应该不知道该数据报是经由归属代理转发而来的。让归属代理将通信者的原始完整数据报**封装**（encapsulate）在一个新的（较大的）数据报中，这两个目标都可以得到满足。这个较大的数据报被导向并交付到移动节点的COA。"拥有"该COA的外部代理将接收并拆封该数据报，即从较大的封装数据报中取出通信者的原始数据报，然后再向移动节点转发该原始数据报（图7-24中的步骤3）。图7-25显示了如下过程：一个通信者向归属网络发送原始数据报；向外部代理发送一个封

装的数据报；以及向移动节点交付最初的数据报。思维敏锐的读者将会注意到，这里描述的封装/拆封概念等同于隧道的概念，隧道是在4.3节讨论IP多播与IPv6时涉及的。

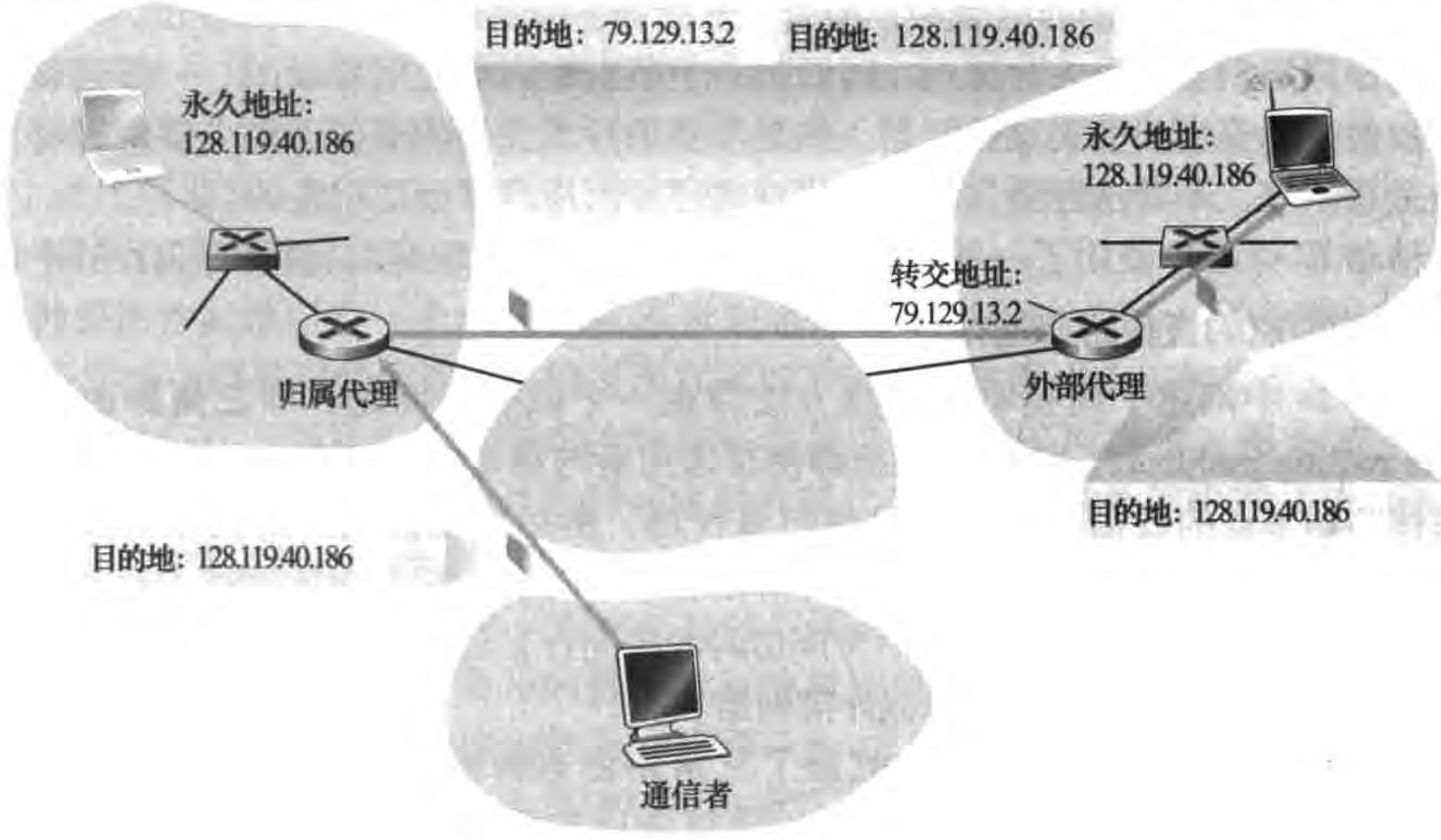


图7-25 封装与拆封

接下来我们考虑某移动节点如何向一个通信者发送数据报。这相当简单，因为移动节点可直接将其数据报寻址到通信者（使用自己的永久地址作为源地址，通信者的地址作为目的地址）。因为移动节点知道通信者的地址，所以没有必要通过归属代理迂回传送数据报。这就是显示在图7-24中的步骤4。

下面通过列出支持移动性所需要的网络层新功能，我们小结一下对有关间接路由选择的讨论。

- *移动节点到外部代理的协议*。当移动节点连接到外部网络时，它向外部代理注册。类似地，当一个移动节点离开该外部网络时，它将向外部代理取消注册。
- *外部代理到归属代理的注册协议*。外部代理将向归属代理注册移动节点的COA。当某移动节点离开其网络时，外部代理不需要显式地注销COA，因为当移动节点移动到一个新网络时，随之而来就要注册一个新的COA，这将完成了注销。
- *归属代理数据报封装协议*。将通信者的原始数据报封装在一个目的地址为COA的数据报内，并转发之。
- *外部代理拆封协议*。从封装好的数据报中取出通信者的原始数据报，然后再将该原始数据报转发给移动节点。

上述讨论提供了一个移动节点在网络之间移动时要维持一个不间断的连接所需的各部分：外部代理、归属代理和间接转发。举一个例子来说明这些部分是如何协同工作的。假设某移动节点连到外部网络A，向其归属代理注册了网络A中的一个COA，并且正在接收通过归属代理间接路由而来的数据报。该移动节点现在移动到外部网络B中，并向网络B中的外部代理注册，外部代理将该移动节点的新COA告诉了其归属代理。此后，归属代理将数据报重路由到网络B。就一个通信者关心的东西而言，移动性是透明的，即在移动前后，数据报都是由相同的归属代理进行路由选择。就归属代理关心的东西而言，数据报流没有中断，即到达的数据报先是转发到外部网络A；改变COA后，则数据报转发到外

部网络 B。但当移动节点在网络之间移动时，它会看到数据报流中断吗？只要移动节点与网络 A 断开连接（此时它不能再经 A 接收数据报）再连接到网络 B（此时它将向归属代理注册一个新的 COA）用的时间少，那么几乎没有丢失数据报。第 3 章讲过，端到端连接可能会由于网络拥塞而丢失数据报。因而当一个节点在网络之间移动时，一条连接中的数据报偶尔丢失算不上什么灾难性问题。如果需要进行无丢失的通信，则上层机制将对数据报丢失进行恢复，不管这种丢失是因网络拥塞还是因用户移动而引发的。

在移动 IP 标准中使用了一种间接路由选择方法［RFC 5944］，这将在 7.6 节中讨论。

2. 移动节点的直接路由选择

在图 7-24 中阐述了间接路由选择方法存在一个低效的问题，即**三角路由选择问题**（triangle routing problem）。该问题是指即使在通信者与移动节点之间存在一条更有效的路由，发往移动节点的数据报也要先发给归属代理，然后再发送到外部网络。在最坏情况下，设想一个移动用户正在访问一位同行所在的外部网络，两人并排坐在一起且正在通过网络交换数据。从通信者（在该例中为该访问者的同行）处发出的数据报被路由选择到该移动用户的归属代理，然后再回到该外部网络！

直接路由选择（direct routing）克服了三角路由选择的低效问题，但却是以增加复杂性为代价的。在直接路由选择方法中，通信者所在网络中的一个**通信者代理**（correspondent agent）先知道该移动节点的 COA。这可以通过让通信者代理向归属代理询问得知，这里假设与间接路由选择情况类似，移动节点具有一个在归属代理注册过的最新的 COA。与移动节点可以执行外部代理的功能相类似，通信者本身也可能执行通信者代理的功能。在图 7-26 中显示为步骤 1 和步骤 2。通信者代理然后将数据报直接通过隧道技术发往移动节点的 COA，这与归属代理使用的隧道技术相类似，参见图 7-26 的步骤 3 和步骤 4。

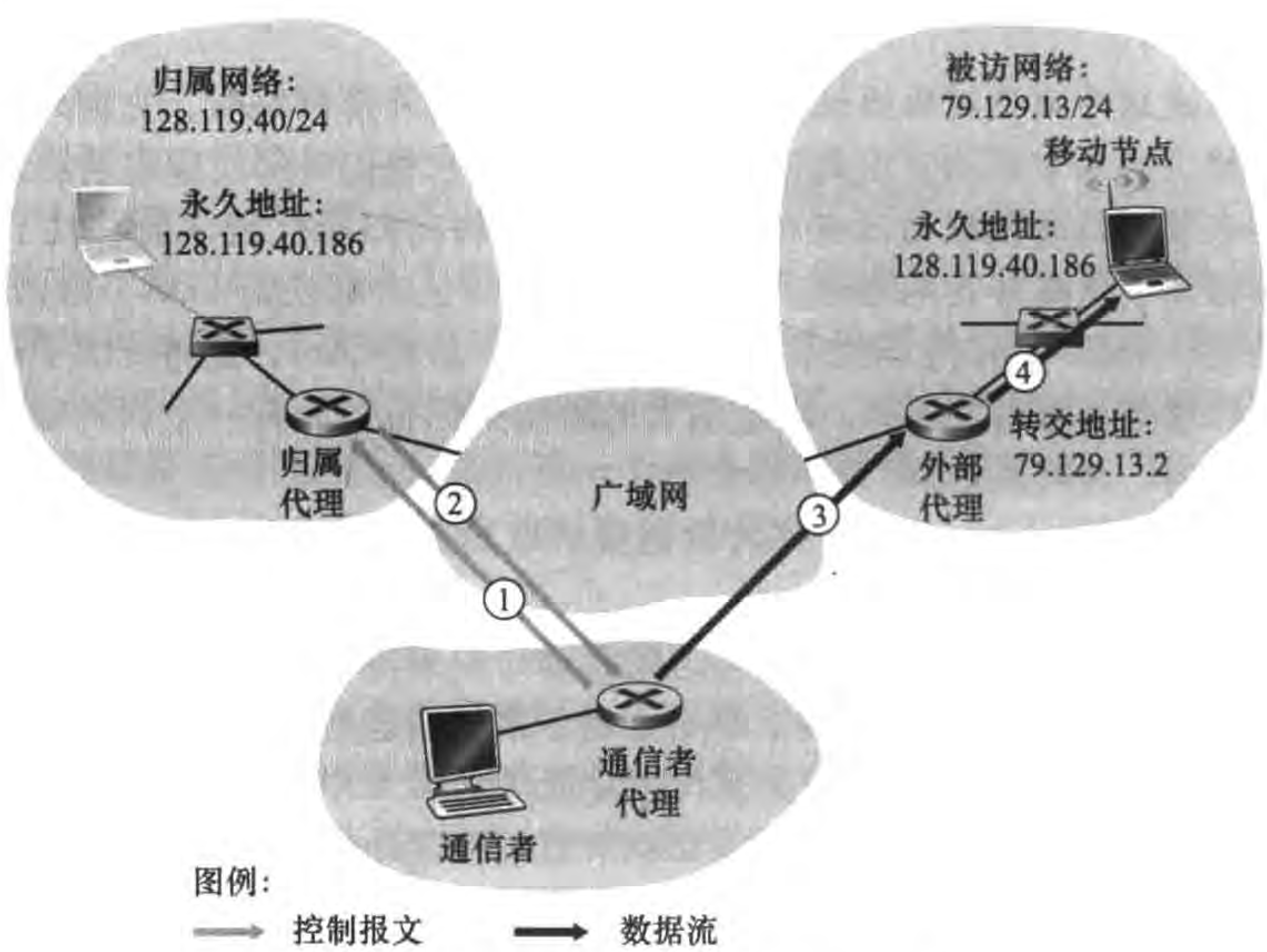


图 7-26　到某移动用户的直接路由选择

尽管直接路由选择克服了三角路由选择问题，但它引入了两个重要的其他挑战：

- 需要一个**移动用户定位协议**（mobile-user location protocol），以便通信者代理向归属代理查询获得移动节点的 COA（图 7-26 中的步骤 1 和步骤 2）。
- 当移动节点从一个外部网络移到另一个外部网络时，如何将数据报转发到新的外部网络？在间接路由选择的情况下，这个问题可以容易地通过更新由归属代理维持的 COA 来解决。然而，使用直接路由选择时，归属代理仅在会话开始时被通信者代理询问一次 COA。因此，当必要时在归属代理中更新 COA，这并不足以解决将数据路由选择到移动节点新的外部网络的问题。

一种解决方案是创建一个新的协议来告知通信者变化后的 COA。另一种方案也是在 GSM 网络实践中所采用的方案，它的工作方式如下。假设数据当前正转发给位于某个外部网络中的移动节点，并且在会话刚开始时该移动节点就位于该网络中（图 7-27 中的步骤 1）。我们将首次发现移动节点的外部网络中的外部代理标识为**锚外部代理**（anchor foreign agent）。当移动节点到达一个新外部网络后（图 7-27 中的步骤 2），移动节点向新的外部代理注册（步骤 3），并且新外部代理向锚外部代理提供移动节点的新 COA（步骤 4）。当锚外部代理收到一个发往已经离开的移动节点的封装数据报后，它可以使用新的 COA 重新封装数据报并将其转发给该移动节点（步骤 5）。如果移动节点其后又移到另一个外部网络中，在该被访网络中的外部代理随后将与锚外部代理联系，以便建立到该新外部网络的转发。

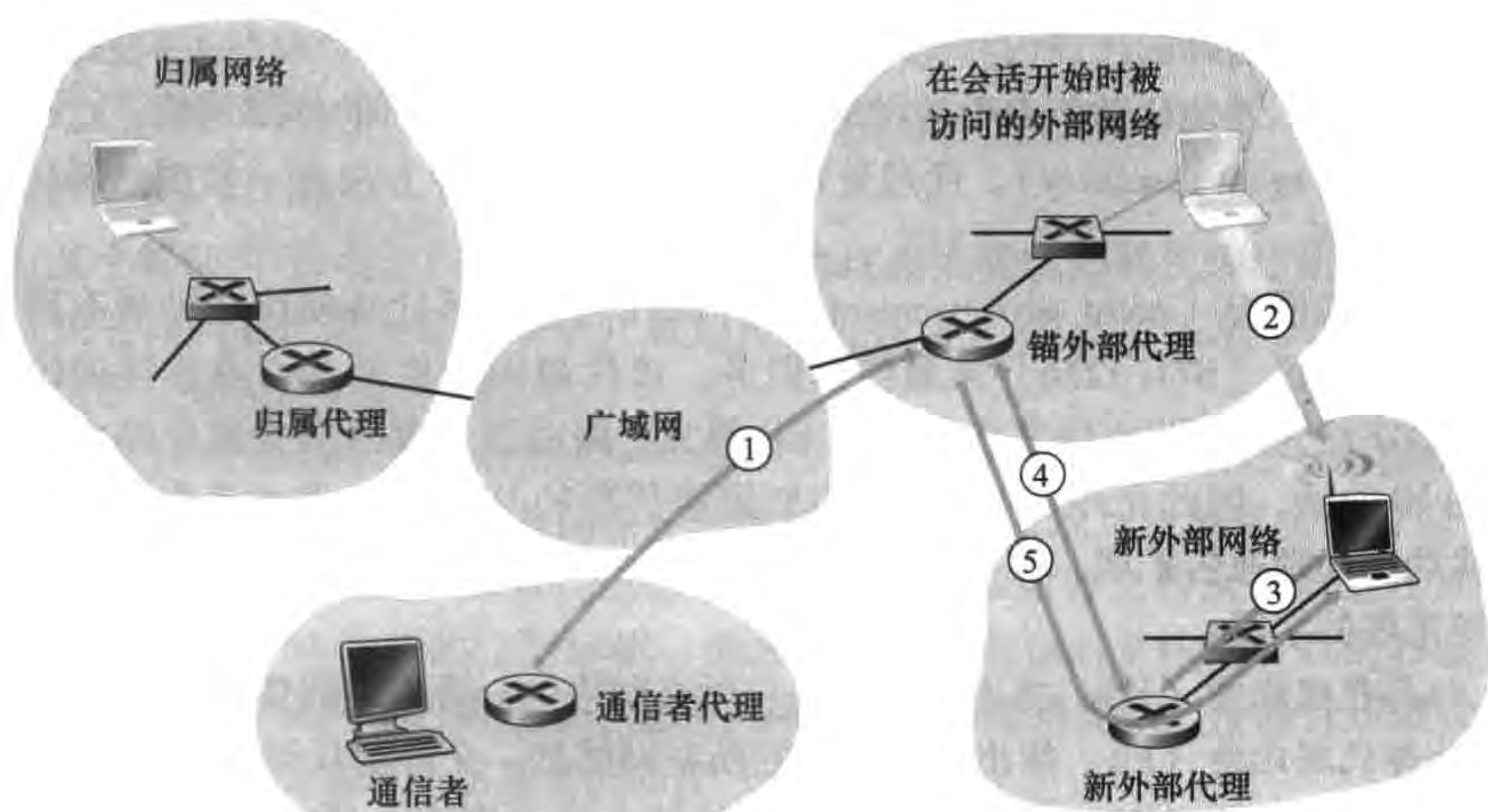


图 7-27 在网络间使用直接路由选择的移动转移

7.6 移动 IP

支持移动性的因特网体系结构与协议合起来称为移动 IP，对 IPv4 主要由 RFC 5944 定义。移动 IP 是一个灵活的标准，支持许多不同的运行模式（例如，具有或不具有外部代理的运行），代理与移动节点相互发现的多种方式，使用单个或多个 COA，以及多种形式的封装。同样，移动 IP 是一个复杂的标准，需要用整本书才能详细描述；的确有这样一本书 [Perkins 1998b]。这里，我们最基本的目标是对移动 IP 最重要的部分进行概述，并说明它在一些常见情形中的使用。

移动 IP 体系结构包含了许多我们前面考虑过的要素，包括归属代理、外部代理、转交地址和封装/拆封等概念。当前的标准［RFC 5944］规定到移动节点使用间接路由选择的方法。

移动 IP 标准由三部分组成：

- 代理发现。移动 IP 定义了一个归属代理或外部代理用来向移动节点通告其服务的协议，以及移动节点请求一个外部代理或归属代理的服务所使用的协议。
- 向归属代理注册。移动 IP 定义了移动节点和/或外部代理向一个移动节点的归属代理注册或注销 COA 所使用的协议。
- 数据报的间接路由选择。该标准也定义了数据报被一个归属代理转发给移动节点的方式，包括转发数据报使用的规则、处理差错情况的规则和几种不同的封装形式［RFC 2003；RFC 2004］。

在整个移动 IP 标准中安全性的考虑是很重要的。例如，显然需要对一个移动节点进行鉴别以确保一个恶意用户不能向归属代理注册一个伪造的转交地址，伪造地址将导致所有发给某个 IP 地址的数据报被重定向到恶意用户。移动 IP 使用许多机制来实现安全性，我们将在第 8 章考察这些机制，在以下的讨论中将不考虑安全性问题。

1. 代理发现

到达一个新网络的某移动 IP 节点，不管是连到一个外部网络还是返回其归属网络，它都必须知道相应的外部代理或归属代理的身份。的确，这是新外部代理的发现，通过一个新的网络地址，才使移动节点中的网络层知道它已进入一个新的外部网络。这个过程被称为**代理发现**（agent discovery）。代理发现可以通过下列两种方法之一实现：经代理通告或者经代理请求。

借助于**代理通告**（agent advertisement），外部代理或归属代理使用一种现有路由器发现协议的扩展协议［RFC 1256］来通告其服务。该代理周期性地在所有连接的链路上广播一个类型字段为 9（路由器发现）的 ICMP 报文。路由器发现报文也包含路由器（即该代理）的 IP 地址，因此允许一个移动节点知道该代理的 IP 地址。路由器发现报文还包括了一个移动性代理通告扩展，其中包含了该移动节点所需的附加信息。在这种扩展中有如下一些较重要的字段：

- 归属代理比特（H）。指出该代理是它所在网络的一个归属代理。
- 外部代理比特（F）。指出该代理是它所在网络的一个外部代理。
- 注册要求比特（R）。指出在该网络中的某个移动用户必须向某个外部代理注册。特别是，一个移动用户不能在外部网络（如使用 DHCP）中获得一个转交地址，并假定由它自己承担外部代理的功能，无须向外部代理注册。
- M、G 封装比特。指出除了“IP 中的 IP”（IP-in-IP）封装形式外，是否还要用其他的封装形式。
- 转交地址（COA）字段。由外部代理提供的一个或多个转交地址的列表。在下面的例子中，COA 将与外部代理关联，外部代理将接收发给该 COA 的数据报，然后再转发到适当的移动节点。移动用户在向其归属代理注册时将选择这些地址中的一个作为其 COA。

图 7-28 说明了在代理通告报文中的某些关键字段。

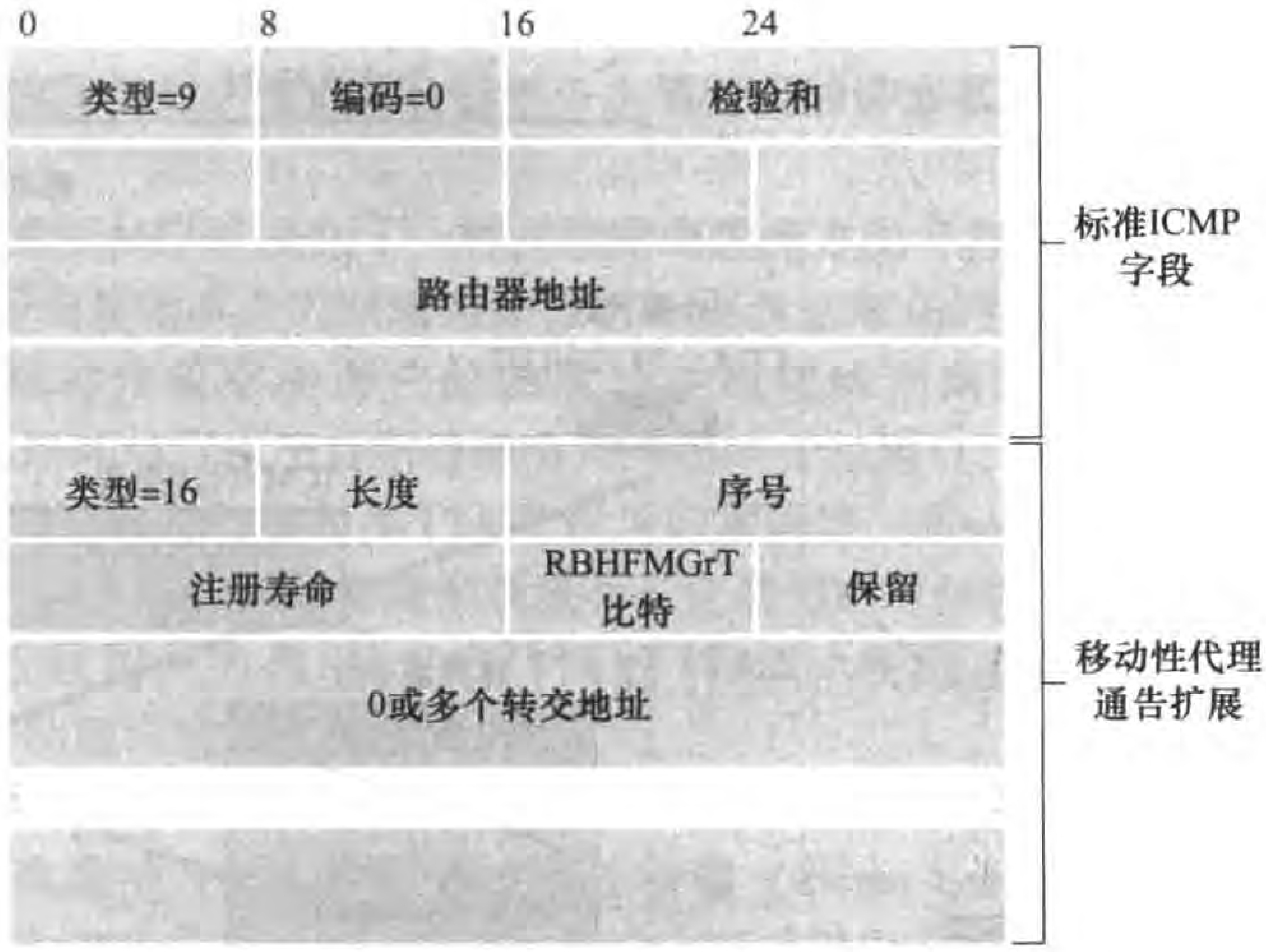


图 7-28 具有移动性代理通告扩展的 ICMP 路由器发现报文

使用**代理请求**（agent solicitation），一个想知道代理的移动节点不必等待接收代理通告，就能广播一个代理请求报文，该报文只是一个类型值为 10 的 ICMP 报文。收到该请求的代理将直接向该移动节点单播一个代理通告，于是该移动节点将继续处理，就好像刚收到一个未经请求的通告一样。

2. 向归属代理注册

一旦某个移动 IP 节点收到一个 COA，则该地址必须要向归属代理注册。这可通过外部代理（由它向归属代理注册该 COA）或直接通过移动 IP 节点自己来完成。我们下面考虑前一种情况，共涉及 4 个步骤。

1）当收到一个外部代理通告后，一个移动节点立即向外部代理发送一个移动 IP 注册报文。注册报文承载在一个 UDP 数据报中并通过端口 434 发送。注册报文携带以下内容：一个由外部代理通告的 COA、归属代理的地址（HA）、移动节点的永久地址（MA）、请求的注册寿命和一个 64 比特的注册标识。请求的注册寿命指示了注册有效的秒数。如果注册没有在规定的时间内在归属代理上更新，则该注册将变得无效。注册标识就像一个序号，用于收到的注册回答与注册请求的匹配（下面会讨论）。

2）外部代理收到注册报文并记录移动节点的永久 IP 地址。外部代理知道现在它应该查找这样的数据报，即它封装的数据报的目的地址与该移动节点的永久地址相匹配。外部代理然后向归属代理的 434 端口发送一个移动 IP 注册报文（同样封装在 UDP 数据报中）。这一报文包括 COA、HA、MA、封装格式要求、请求的注册寿命以及注册标识。

3）归属代理接收注册请求并检查真实性和正确性。归属代理把移动节点的永久 IP 地址与 COA 绑定在一起。以后，到达该归属代理的数据报与发往移动节点的数据报将被封装并以隧道方式给 COA。归属代理发送一个移动 IP 注册回答，该响应报文中包含有 HA、MA、实际注册寿命和被认可的请求报文注册标识。

4）外部代理接收注册响应，然后将其转发给移动节点。

到此，注册便完成了，移动节点就能接收发送到其永久地址的数据报。图 7-29 说明了这些步骤。注意到归属代理指定的寿命比移动节点请求的寿命要小。

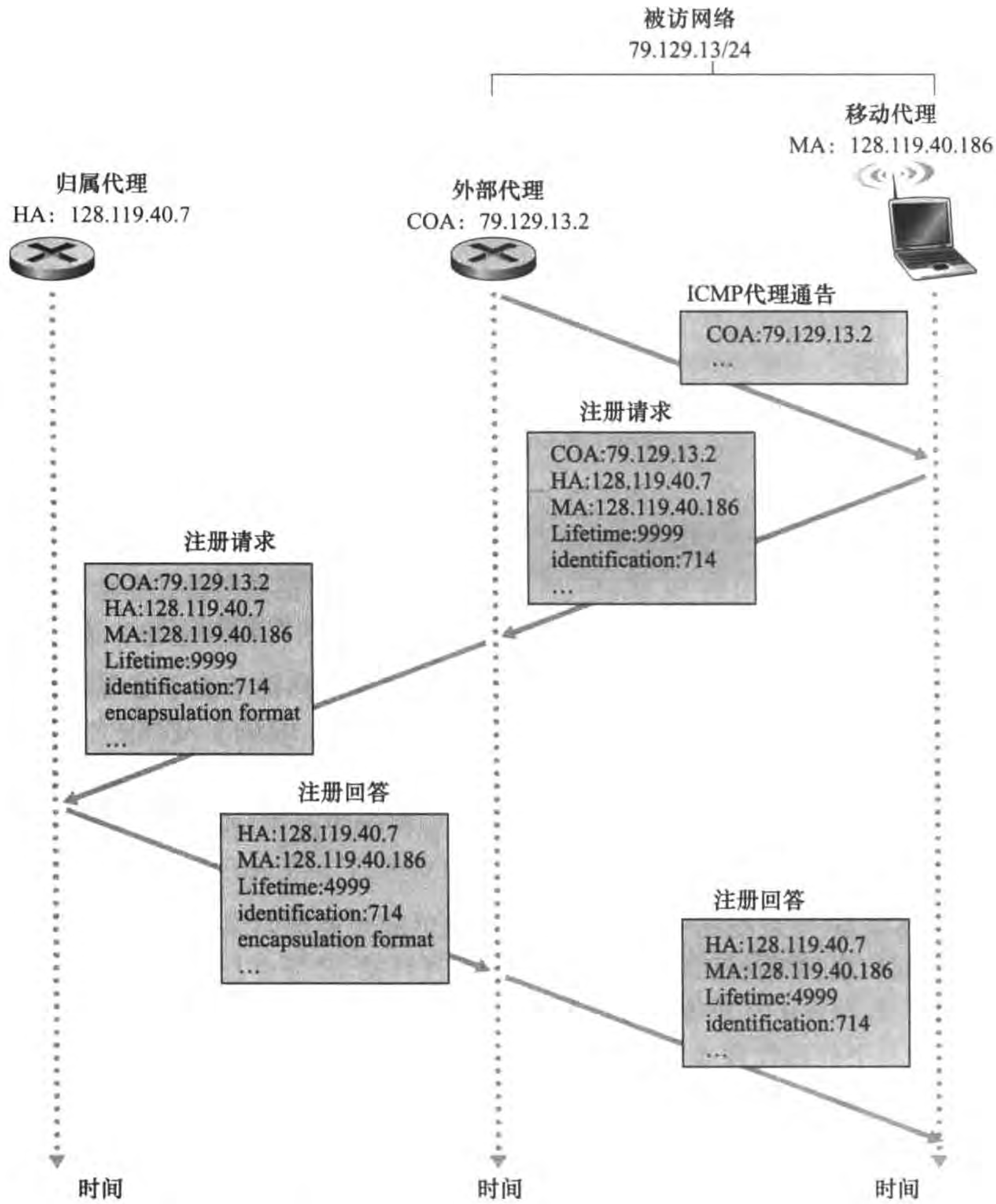


图7-29 代理通告与移动IP注册

当某个移动节点离开其网络时，外部代理无须显式地取消某个COA的注册。当移动节点移动到一个新网（不管是另一个外部网络还是其归属网络）并注册一个新COA时，上述情况将自动发生。

除了上面所描述的情况，移动IP标准还允许许多另外的情形和功能，有兴趣的读者可以参阅［Perkins 1998b；RFC 5944］。

7.7 管理蜂窝网中的移动性

分析了IP网络中的移动性管理以后，我们现将注意力转向对移动性支持有更长历史的网络，即蜂窝电话网络。尽管在7.4节中我们关注过蜂窝网中的第一跳无线链路，但这里我们关注移动性，并以GSM蜂窝网络体系结构［Goodman 1997；Mouly 1992；Scourias 2012；Kaaranen 2001；Korhonen 2003；Turner 2012］作为学习案例，因为它是一个成熟并

被广泛部署的技术。3G 和 4G 网络中的移动性原则上与 GSM 中所使用的移动性类似。与在移动 IP 中的情况类似，我们将会看到 7.5 节指出的许多基本原理都被包含在 GSM 网络体系结构中。

与移动 IP 类似，GSM 采用了一种间接路由选择方法（参见 7.5.2 节），首先将通信者的呼叫路由选择到移动节点的归属网络，再从那到达被访网络。在 GSM 术语中，移动用户的归属网络被称作该移动用户的**归属公共地域移动网络**（home Public Land Mobile Network，home PLMN）。由于首字母缩略词 PLMN 有些拗口，考虑到我们避免缩略词字母表的要求，我们直接将 GSM 归属 PLMN 称为**归属网络**（home network）。移动用户向某个蜂窝网提供商订购了服务，该蜂窝网就成为了这些用户的归属网络（即该提供商就按月提供的蜂窝服务收取用户的费用）。被访问的 PLMN，我们直接称其为**被访网络**（visited network），是移动用户当前所在网络。

与移动 IP 中情况类似，归属网络和被访网络的职责有很大的差别。

- 归属网络维护一个称作**归属位置注册器**（Home Location Register，HLR）的数据库，其中包括它每个用户的永久蜂窝电话号码以及用户个人概要信息。重要的是，HLR 也包括这些用户当前的位置信息。这就是说，如果一个移动用户当前漫游到另一个提供商的蜂窝网络中，HLR 中将包含足够多的信息来获取（通过一个我们即将描述的过程）被访网络中对移动用户的呼叫应该路由选择到的地址。我们将会看到，当一个呼叫定位到一个移动用户后，通信者将与归属网络中一个被称作**网关移动服务交换中心**（Gateway Mobile services Switching Center，GMSC）的特殊交换机联系。同样，为避免拗口的缩略词，我们这里用一个更具描述性的术语来称呼 GMSC，即**归属 MSC**（home MSC）。
- 被访网络维护一个称作**访问者位置注册**（Visitor Location Register，VLR）的数据库。VLR 为每一个当前在其服务网络中的移动用户包含一个表项，VLR 表项因此随着移动用户进入和离开网络而出现或消失。VLR 通常与移动交换中心（MSC）在一起，该中心协调到达或离开被访网络的呼叫建立。

在实践中，一个服务商的蜂窝网络将为其用户提供归属网络服务，同时为在其他蜂窝服务商订购服务的移动用户提供被访网络服务。

7.7.1 对移动用户呼叫的路由选择

现在我们描述一个呼叫如何定位到被访网络中的一个移动 GSM 用户。我们首先考虑下面一个简单的例子，更复杂的例子在 [Mouly 1992] 中有描述。如图 7-30 所示，这些步骤如下：

1）通信者拨打移动用户的电话号码。该号码本身并不涉及一个特定的电话线路或位置（毕竟电话号码是固定的，而用户是移动的），号码中的前几位数字足以全局地判别移动用户的归属网络。呼叫从通信者通过公共交换电话网到达移动用户归属网络中的归属 MSC。这是呼叫的第一步。

2）归属 MSC 收到该呼叫并查询 HLR 来确定移动用户的位置。在最简单的情况下，HLR 返回**移动站点漫游号码**（Mobile Station Roaming Number，MSRN），我们称其为**漫游号码**（roaming number）。注意到这个号码与移动用户的永久电话号码不同，后者是与移动用户的归属网络相关联的，而漫游号码是短暂的：当移动用户进入一个被访网络后，会给移动用户临时分配一个漫游号码。漫游号码的作用就相当于移动 IP 中转交地址的作用。并

且，与COA类似，它也是对通信者和移动用户不可见的。如果HLR不具有该漫游号码，它返回被访网络中VLR的地址。在这种情况下（未在图7-30中显示出来），归属MSC需要查询VLR以便获取移动节点的漫游号码。但是HLR是如何首先得到漫游号码或VLR地址的呢？移动用户到另一个被访网络后这些值将发生怎样的变化？我们将很快考虑这些重要问题。

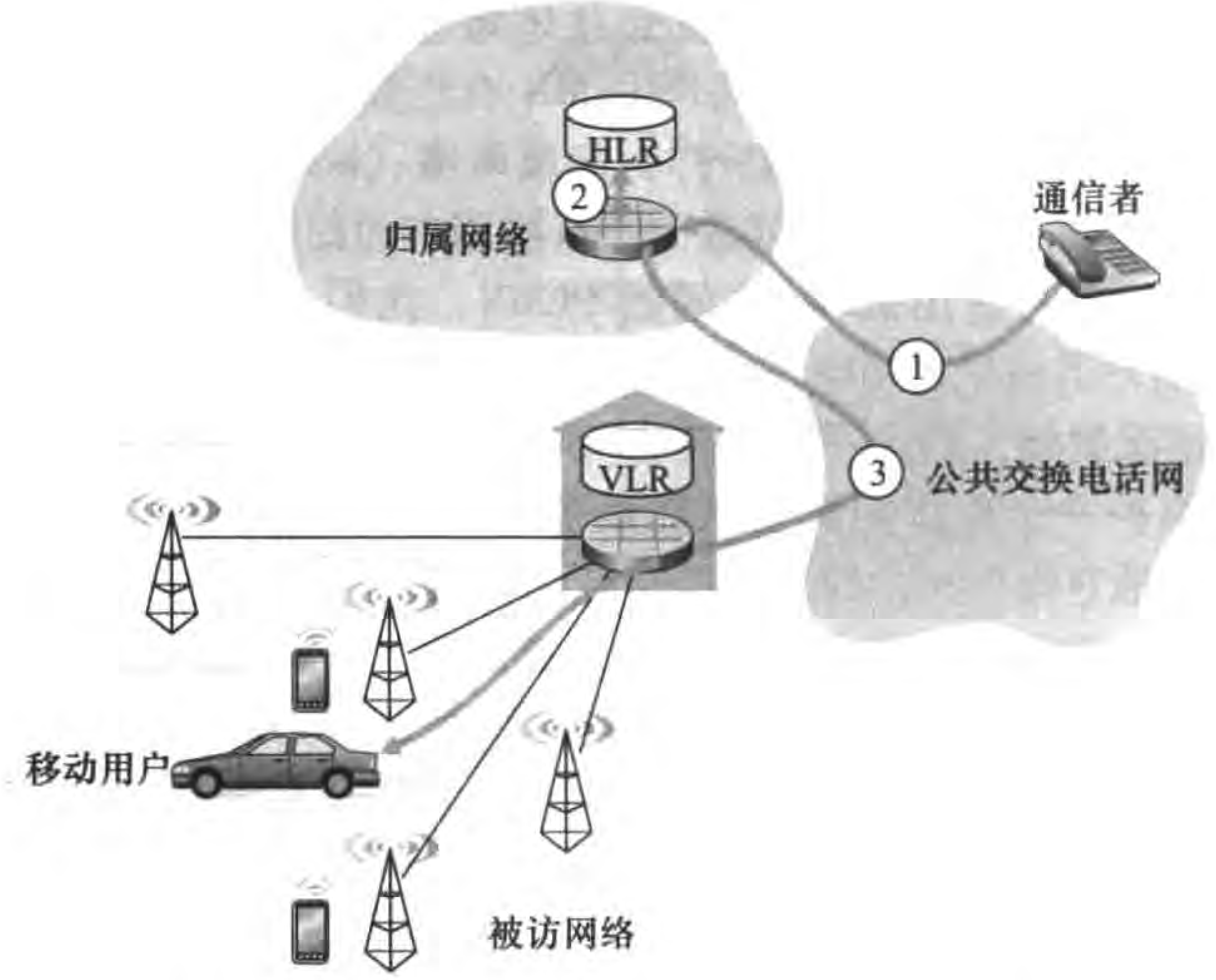


图7-30　将呼叫定位到一个移动用户：间接路由选择

3）给定一个漫游号码，归属MSC通过网络到达被访网络的MSC建立呼叫的第二步。至此，该呼叫已经完成，从通信者到达归属MSC，再从归属MSC到达被访MSC，然后到达为移动用户提供服务的基站。

在第二步中，一个未解决的问题是HLR如何获得有关移动用户位置的信息。当一个移动电话切换或进入一个由新的VLR所覆盖的被访网络中以后，移动用户必须向被访网络注册，这是通过在移动用户和VLR之间交换信令报文来实现的。被访VLR随后又向移动用户的HLR发送一个位置更新请求报文。这一报文告知HLR可以用来联系移动用户的漫游号码，或者VLR地址（它可以用来随后查询以获取移动号码）。作为这个交换的一部分，VLR同样从HLR那里获取移动用户的信息，以及确定被访网络应该给予移动用户什么样的服务（如果有的话）。

7.7.2　GSM中的切换

在一个呼叫过程中，移动站点将其关联从一个基站改变到另一个基站时出现**切换**（hand-off）。如图7-31所示，移动用户的呼叫初始时（在切换前）通过一个基站（我们称其为旧基站）路由选择到该移动用户，而在切换以后它经过另一个基站（我们称其为新基站）路由选

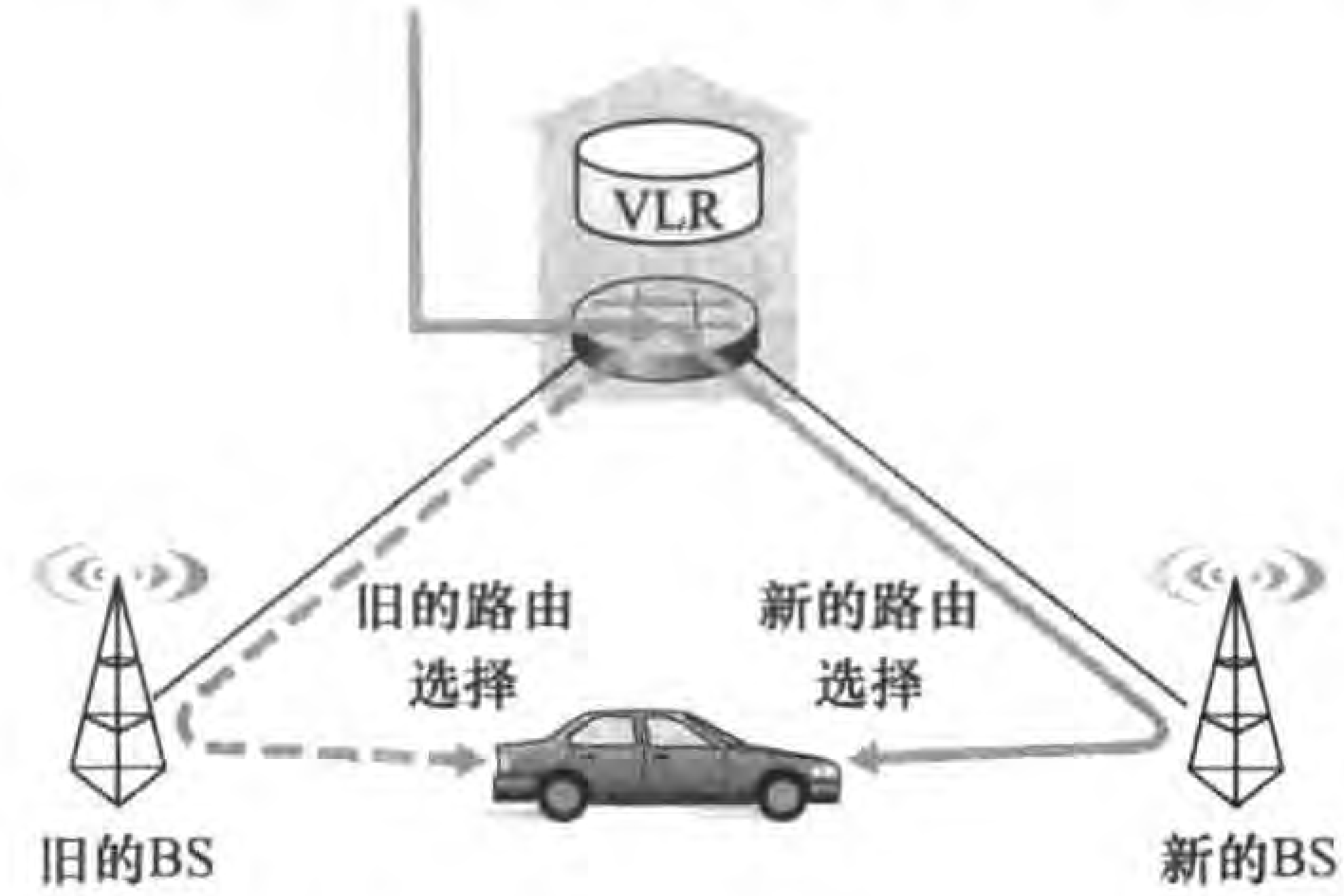


图7-31　具有一个公共MSC的基站间的切换情况

择到移动用户。注意到基站之间的切换不仅导致移动用户向/从一个新的基站传输/接收信号，而且导致正在进行的呼叫从网络中的一个交换点到新基站的重路由选择。我们首先假设新旧基站共享同一个 MSC，并且重路由选择发生在这个 MSC。

有几种原因导致切换的发生，包括：①当前基站和移动用户之间的信号减弱，使得该呼叫有被中断的危险；②一个蜂窝处理的呼叫太多，变得过载。可以通过将一些移动用户切换到邻近不太拥塞的蜂窝中，使这个拥塞得到缓解。

在与一个基站相关联期间，移动用户周期性地测量来自其当前基站和临近它的可以“听得到”的基站的信标信号强度。这些测量以每秒 1 ~2 次的频率报告给移动用户的当前基站。根据这些测量值、临近蜂窝的移动用户的当前负载以及其他因素，GSM 中的切换由旧的基站发起［Mouly 1992］。GSM 标准并未明确规定基站在确定是否进行切换时所采用的具体算法。

图 7-32 显示了当一个基站决定切换一个移动用户时所包括的步骤：

1）旧基站（BS）通知被访问 MSC 即将要进行一个切换，通知移动用户将要切换到的 BS（或可能的 BS 集）。

2）被访问 MSC 发起建立到新 BS 的路径，分配承载重路由选择的呼叫所需的资源，以及用信令告知新 BS 一个切换即将出现。

3）新 BS 分配并激活一个无线信道供移动用户使用。

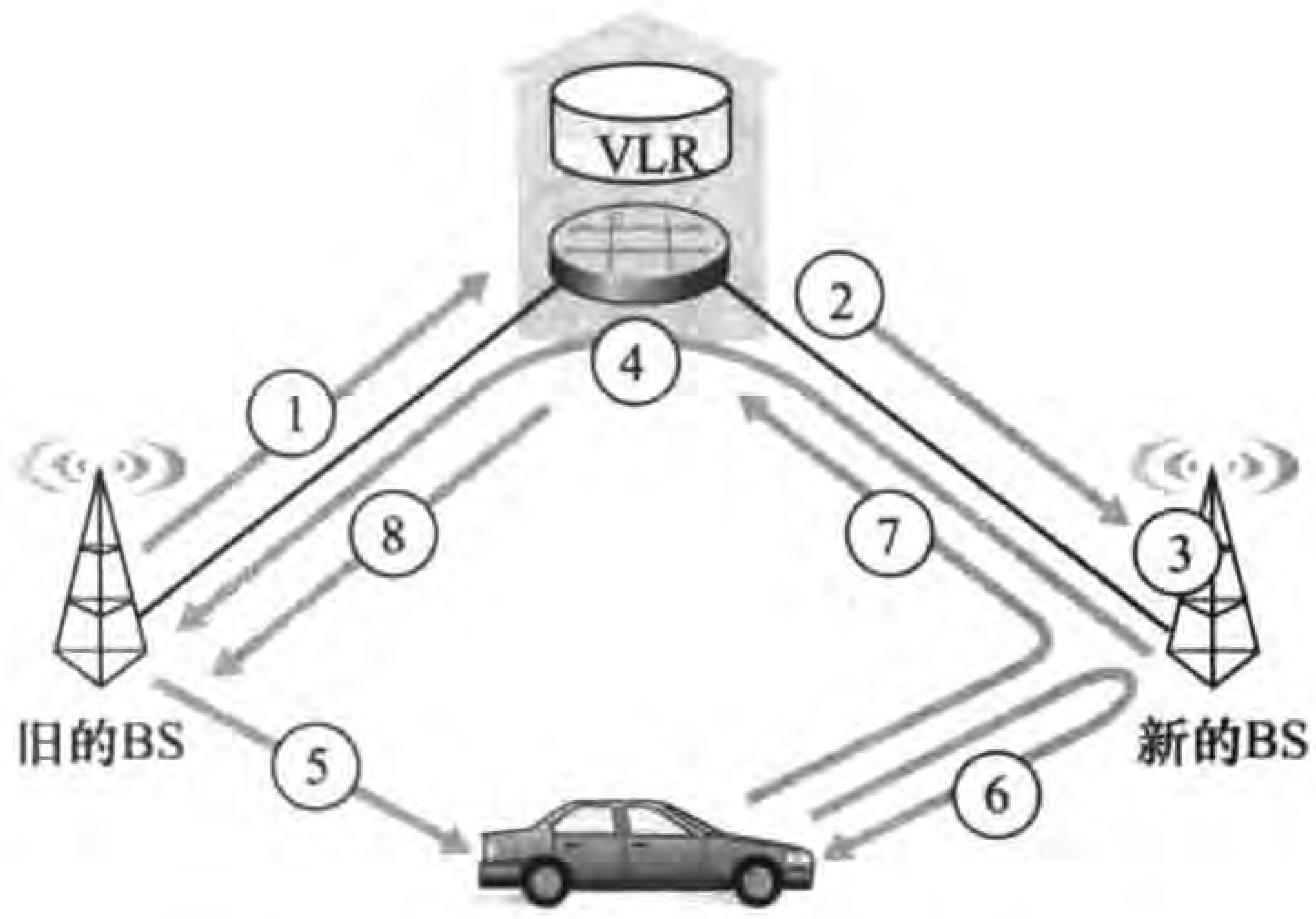


图 7-32　具有一个公共 MSC 的基站间完成一个切换的步骤

4）新 BS 发出信令返回被访问 MSC 和旧 BS，即已经建立了被访问 MSC 到新 BS 的路径并且移动用户应当被告知即将发生的切换。新 BS 提供移动用户与新的 BS 相关联所需要的所有信息。

5）移动用户被告知它应当进行一个切换。注意到此时为止，移动用户完全不知网络已经为切换做好所有底层工作（如在新 BS 中分配一个信道，分配一条从被访问 MSC 到新 BS 的路径）。

6）移动用户和新 BS 交换一个或多个报文，以完全激活新 BS 中新的信道。

7）移动用户向新 BS 发送一个切换完成报文，该报文随后向上转发给被访问 MSC。该被访问 MSC 然后重路由选择到移动用户的正在进行的呼叫，使其经过新 BS。

8）沿着到旧 BS 的路径分配的资源随后被释放。

通过考虑如下情况来总结我们对切换的讨论：当移动用户移动到一个不同于旧 BS 的、与不同的 MSC 关联的 BS 中时，并且当这种 MSC 之间的切换多次发生时，考虑这些情况下将发生什么。如图 7-33 所示，GSM 定义了**锚 MSC**（anchor MSC）的概念。锚 MSC 是呼叫首次开始时移动用户所访问的 MSC，它因此在整个呼叫持续过程中保持不变。在整个呼叫持续期间，不论移动用户进行了多少次 MSC 间转换，呼叫总是从归属 MSC 路由选择到锚 MSC，然后再到移动用户当前所在的被访问 MSC。当移动用户从一个 MSC 覆盖区到达另一个 MSC 覆盖区后，正在进行的呼叫被重路由选择，从锚 MSC 到包含新基站的新被访问 MSC。因此，在任何情况下，通信者和移动用户之间至多有 3 个 MSC（归属 MSC、锚 MSC 以及被访问 MSC）。图 7-33 图示了在移动用户所访问的 MSC 之间的一个呼叫的路由选择。

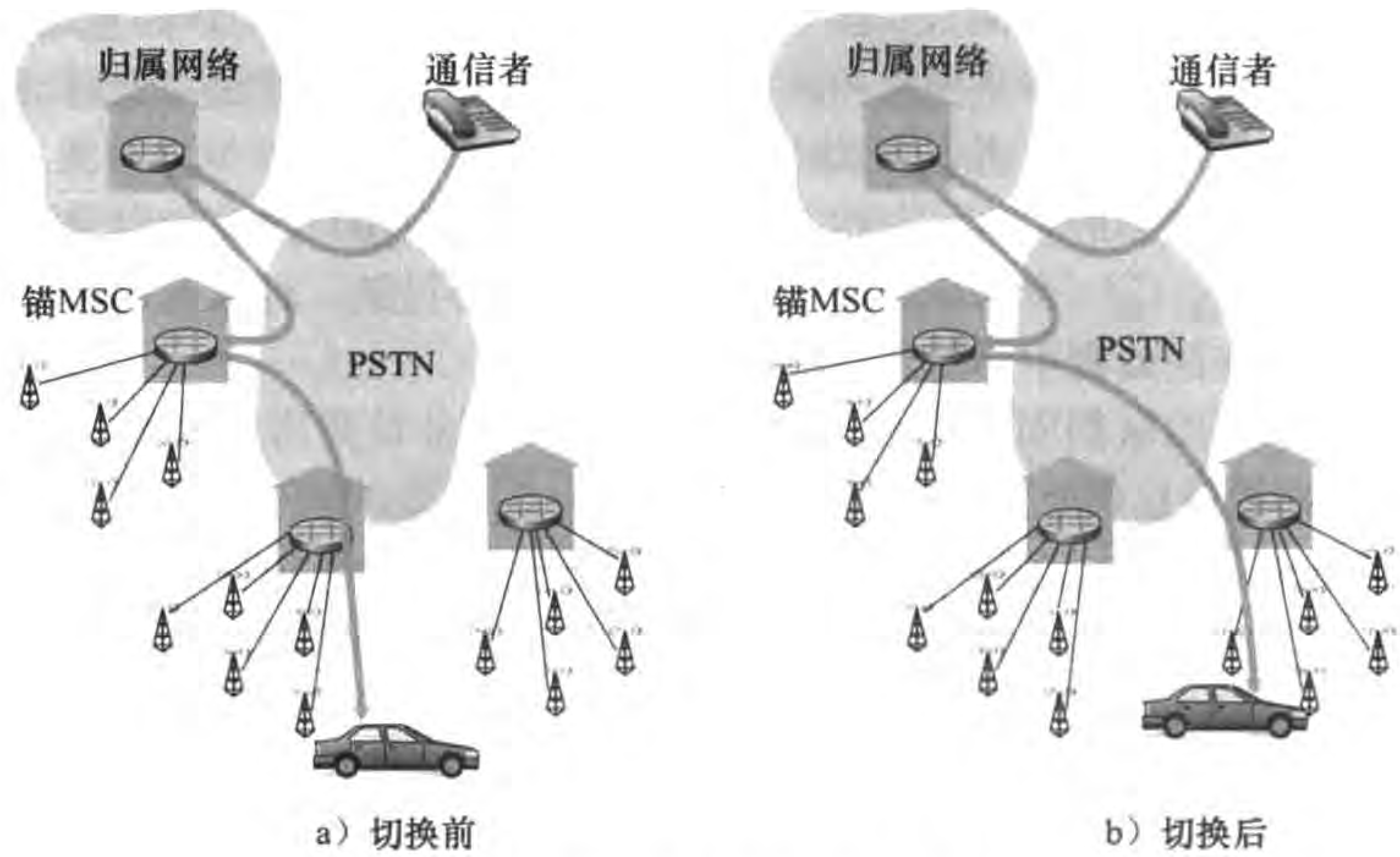


图7-33　通过锚MSC重路由选择

另一种方法则不用维持从锚MSC到当前MSC的单一MSC跳，将直接链接移动用户访问的MSC。每当移动用户移到一个新MSC后，让旧MSC将正在进行的呼叫转发给新MSC。这种MSC链事实上能够出现在IS-41蜂窝网络中，通过使用最少步骤的可选路径来去除在锚MSC和当前访问MSC之间的MSC [Lin 2001]。

下面通过对比GSM和移动IP中的移动性管理，来完成我们对GSM移动性管理的讨论。表7-2中的对比指出了尽管IP和蜂窝网络在很多方面有很大的区别，但它们共享数量惊人的公共功能要素和处理移动性的总体方法。

表7-2　移动IP和GSM移动性之间的共性

GSM要素	对GSM要素的解释	移动IP要素
归属系统	移动用户永久电话号码所归属的网络	归属网络
网关移动（服务）交换中心或简称归属MSC，归属位置注册器（HLR）	归属MSC：获取移动用户路由地址的联系点。HLR：归属系统中包含移动用户永久电话号码、个人信息、当前位置和订购信息的数据库	归属代理
被访问系统	移动用户当前所在的非归属系统网络	被访网络
被访问移动（服务）交换中心或简称被访问MSC，访问者定位记录（VLR）	被访问MSC：负责建立与MSC相关联的发射区中到/从移动节点的呼叫。VLR：访问系统中的临时数据库项，包含每个访问移动用户的订购信息	外部代理
移动站点漫游号码（MSRN），或漫游号码	用于归属MSC和被访问MSC之间电话呼叫的路由地址，对移动用户和通信者均不可见	转交地址

7.8　无线和移动性：对高层协议的影响

在本章中，我们已经看到了无线网络在链路层（由于无线信道的诸如衰减、多径、隐终端等特性）和网络层（由于移动用户改变与网络的连接点）与有线网络的对应物有重大的区别。但在运输层和应用层是否也有重大差别呢？我们很容易认为这些差别是很小的，因为在有线和无线网络中的网络层均为上层提供了同样的尽力而为服务模式。类似

地，如果在有线和无线网络中都是使用诸如 TCP 和 UDP 的协议提供运输层服务，那么应用层也应该保持不变。在某个方面我们的直觉是对的，即 TCP 和 UDP 可以（也确实）运行在具有无线链路的网络中。在另一方面，运输层协议（特别是 TCP）通常在有线和无线网络中有时会有完全不同的性能。这里，在性能方面区别是明显的，我们来研究一下其中的原因。

前面讲过，在发送方和接收方之间的路径上，一个报文段不论是丢失还是出错，TCP 都将重传它。在移动用户情况下，丢失可能源于网络拥塞（路由器缓存溢出）或者切换（例如，由于重路由选择报文段到移动用户新的网络接入点引入的时延）。在所有情况下，TCP 的接收方到发送方的 ACK 都仅仅表明未能收到一个完整的报文段，发送方并不知道报文段是由于拥塞，或在切换过程中，还是由于检测到比特差错而被丢弃的。在所有情况下，发送方的反应都一样，即重传该报文段。TCP 的拥塞控制响应在所有场合也是相同的，即 TCP 减小其拥塞窗口，如 3.7 节讨论的那样。由于无条件地降低其拥塞窗口，TCP 隐含地假设报文段丢失是由于拥塞而非出错或者切换所致。我们在 7.2 节看到，在无线网络中比特错误比在有线网络中普遍得多。当这样的比特差错或者切换丢失发生时，没理由让 TCP 发送方降低其拥塞窗口（并因此降低发送速率）。此时路由器的缓存的确可能完全是空的，分组可以在端到端链路中丝毫不受拥塞阻碍地流动。

研究人员在 20 世纪 90 年代早期到中期就认识到，由于无线信道的高比特差错率和切换丢失的可能性，TCP 的拥塞控制反应在无线情况下可能会有问题。有三大类可能的方法用于处理这一问题：

- *本地恢复*。本地恢复方法的目标是在比特差错出现的当时和当地（如在无线链路中）将其恢复。如在 7.3 节学习的 802.11 ARQ 协议，或者使用 ARQ 和 FEC 的更为复杂的方法 [Ayanoglu 1995]）。
- *TCP 发送方知晓无线链路*。在本地恢复方法中，TCP 发送方完全不清楚其报文段跨越一段无线链路。另一种方法是让 TCP 发送方和接收方知道无线链路的存在，从而将在有线网络中发生的拥塞性丢包和在无线网络中发生的差错/丢包区分开，并且仅对有线网络中的拥塞性丢包采用拥塞控制。在假设端系统能够做出这种区分的情况下，[Balakrishnan 1997] 详细研究了多种类型的 TCP。[Liu 2003] 研究了在一个端到端路径中区分有线部分丢包和无线部分丢包的技术。
- *分离连接方法*。在分离连接方法中 [Bakre 1995]，移动用户和其他端点之间的端到端连接被打断为两个运输层连接：一个从移动主机到无线接入点，一个从无线接入点到其他通信端点（我们假定它是有线的主机）。该端到端连接因此是由一个无线部分和一个有线部分级连形成的。经无线段的运输层能够是一个标准的 TCP 连接 [Bakre 1995]，或是一个特别定制运行在 UDP 上的差错恢复协议。[Yavatkar 1994] 研究了经无线连接使用运输层选择性重传协议。在 [Wei 2006] 中的测量报告指出了分离 TCP 连接广泛用于蜂窝数据网络中，通过使用分离 TCP 连接，上述问题的确能够有很大改进。

我们这里有关无线链路上的 TCP 的讨论是十分简要的。在无线网络中有关 TCP 挑战和解决方案的深入展望能够在 [Hanabali 2005；Leung 2006] 中找到。我们鼓励读者去查阅这些文献以了解这个正在进行的研究领域的详情。

考虑过运输层协议后，我们接下来考虑无线和移动性对应用层协议的影响。这里一个重要的考虑是无线链路经常具有相对较低的带宽，如我们在图 7-2 中所见。因此，运行在

无线链路尤其是蜂窝无线链路上的应用程序，必须将带宽作为稀有物品对待。例如，一个为在4G电话上运行的Web浏览器提供服务的Web服务器，就不能像为运行在有线连接的浏览器那样提供含有大量图片的内容。尽管无线链路的确为应用层提出一些挑战，它们具有的移动性同样使得一大批位置知晓和环境知晓应用成为可能［Chen 2000；Baldauf 2007］。更一般地，无线和移动网络将在未来的泛在计算环境实现中起着重要作用［Weiser 1991］。显然，在谈及无线和移动网络对网络应用及其协议的影响时，公平而论我们仅看到了冰山一角！

7.9 小结

无线网络和移动网络使电话发生了革命性变化，同时也对计算机网络界产生了日益深远的影响。伴随着它们对全球网络基础设施的随时、随地、无缝地接入，它们不仅使网络接入变得更加无所不在，而且催生了一组新的、令人兴奋的位置相关服务。考虑到无线网络和移动网络不断增长的重要性，本章关注用于支持无线和移动通信的原理、通用链路技术以及网络体系结构。

本章以对无线网络和移动网络的介绍开始，描述了由这种网络中通信链路的无线特性所引发的挑战和由这些无线链路带来的移动性之间的重要区别。这使我们能够更好地区分、识别和掌握每个领域中的关键概念。我们首先关注无线通信，在7.2节中考虑了无线链路的特征。在7.3节和7.4节中，我们研究了IEEE 802.11（WiFi）无线LAN标准、两个IEEE 802.15个人域网络（蓝牙和ZigBee），以及3G和4G蜂窝因特网接入。然后我们将注意力转向移动性问题。在7.5节中我们区分了多种形式的移动性，不同的移动性面临不同的挑战，并且看到了不同的解决方案。我们考虑了移动节点的定位和路由选择问题，以及对那些动态地从一个网络接入点移到另一个网络接入点的移动用户的切换问题。在7.6节和7.7节中，我们分别考察了这些问题在移动IP标准和GSM中是如何处理的。最后，我们在7.8节中考虑了无线链路和移动性对运输层协议和网络应用的影响。

尽管我们用了整整一章来学习无线网络和移动网络，但全面探索这个令人兴奋和快速扩展的领域需要一整本书或更多书的篇幅。我们鼓励读者通过查阅在本章中提供的许多参考资料，对这一领域进行更深入的研究。

课后习题和问题

复习题

7.1节

R1. 一个无线网络运行在“基础设施模式”下是什么含义？如果某网络没有运行在基础设施模式下，那么它运行在什么模式下？这种运行模式与基础设施模式之间有什么不同？

R2. 在7.1节的分类法中，所确定的四种无线网络类型各是什么？你已经使用的是这些无线网络类型中的哪一种？

7.2节

R3. 下列类型的无线信道损伤之间有什么区别：路径损耗、多径传播、来自其他源的干扰？

R4. 随着移动节点离开基站越来越远，为了保证传送帧的丢失概率不增加，基站能够采取的两种措施是什么？

7.3~7.4 节

R5. 描述 802.11 中信标帧的作用。

R6. 是非判断：802.11 站在传输一个数据帧前，必须首先发送一个 RTS 帧并收到一个对应的 CTS 帧。

R7. 为什么 802.11 中使用了确认，而有线以太网中却未使用？

R8. 是非判断：以太网和 802.11 使用相同的帧格式。

R9. 描述 RTS 门限值的工作过程。

R10. 假设 IEEE 802.11 RTS 和 CTS 帧与标准的 DATA 和 ACK 帧一样长，使用 CTS 和 RTS 帧还会有好处吗？为什么？

R11. 7.3.4 节讨论了 802.11 移动性，其中无线站点从一个 BSS 到同一子网中的另一个 BSS。当 AP 是通过交换机互连时，为了让交换机能适当地转发帧，一个 AP 可能需要发送一个带有哄骗的 MAC 地址的帧，为什么？

R12. 在某蓝牙网络中的一个主设备和在 802.11 网络中的一个基站之间有什么不同？

R13. 在 802.15.4 ZigBee 标准中超级帧的含义是什么？

R14. 在 3G 蜂窝数据体系结构中，“核心网”的作用是什么？

R15. 在 3G 蜂窝数据体系结构中，RNC 的作用是什么？在蜂窝语音网中 RNC 起什么作用？

R16. 在 4G 体系结构中 eNodeB、MME、P-GW 和 S-GW 的作用是什么？

R17. 3G 和 4G 蜂窝体系结构之间的 3 个重要差别是什么？

7.5~7.6 节

R18. 如果某节点与因特网具有无线连接，则该节点必定是移动的吗？试解释之。假设一个使用膝上型电脑的用户携带电脑绕着她的住所散步，并且总是通过相同的接入点接入因特网。从网络的角度看，这是移动用户吗？试解释之。

R19. 永久地址与转交地址之间的区别是什么？谁指派转交地址？

R20. 考虑经移动 IP 的一条 TCP 连接。是非判断：在通信者和移动主机之间的 TCP 连接阶段经过该移动用户的归属网络，但数据传输阶段直接通过该通信者和移动主机，绕开了归属网络。

7.7 节

R21. 在 GSM 网络中，HLR 和 VLR 的目的是什么？移动 IP 的什么要素类似于 HLR 和 VLR？

R22. 在 GSM 网络中，锚 MSC 的作用是什么？

7.8 节

R23. 为了避免单一无线链路降低一条端到端运输层 TCP 连接的性能，能够采取的三种方法是什么？

习题

P1. 考虑在图 7-5 中单一发送方的 CDMA 例子。如果发送方的 CDMA 码是（1，-1，1，-1，1，-1，1，-1），那么其输出（对于所显示的两个数据比特）是什么？

P2. 考虑图 7-6 中的发送方 2，发送方对信道 $Z_{i,m}^2$ 的输出是什么（在它被加到来自发送方 1 的信号前）？

P3. 假设在图 7-6 中的接收方希望接收由发送方 2 发送的数据。说明通过使用发送方 2 的代码，（经计算）接收方的确能够将发送方 2 的数据从聚合信道信号中恢复出来。

P4. 在两个发送方、两个接收方的场合，给出一个包括 1 和 -1 值的两个 CDMA 编码的例子，不允许两个接收方从两个 CDMA 发送方提取出初始传输的比特。

P5. 假设有两个 ISP 在一个特定的咖啡馆内提供 WiFi 接入，并且每个 ISP 有其自己的 AP 和 IP 地址块。

a. 进一步假设，两个 ISP 都意外地将其 AP 配置运行在信道 11。在这种情况下，802.11 协议是否将完全崩溃？讨论一下当两个各自与不同 ISP 相关联的站点试图同时传输时，将会发生什么情况。

b. 现在假设一个 AP 运行在信道 1，而另一个运行在信道 11。你的答案将会有什么变化？

P6. 在 CSMA/CA 协议的第 4 步，一个成功传输一个帧的站点在第 2 步（而非第 1 步）开始 CSMA/CA 协议。通过不让这样一个站点立即传输第 2 个帧（如果侦听到该信道空闲），CSMA/CA 的设计者是基

于怎样的基本原理来考虑的呢?

P7. 假设一个 802.11b 站点被配置为始终使用 RTS/CTS 序列预约信道。假设该节点突然要发送 1000 字节的数据,并且所有其他站点此时都是空闲的。作为 SIFS 和 DIFS 的函数,并忽略传播时延,假设无比特差错,计算发送该帧和收到确认需要的时间。

P8. 考虑在图 7-34 中显示的情形,其中有 4 个无线节点 A、B、C 和 D。这 4 个节点的无线电覆盖范围显示为其中的椭圆形阴影;所有节点共享相同的频率。当 A 传输时,仅有 B 能听到/接收到;当 B 传输时,A 和 C 能听到/接收到;当 C 传输时,B 和 D 能听到/接收到;当 D 传输时,仅有 C 能听到/接收到。

假定现在每个节点都有无限多的报文要向每个其他节点发送。如果一个报文的目的地不是近邻,则该报文必须要中继。例如,如果 A 要向 D 发送,来自 A 的报文必须首先发往 B,B 再将该报文发送给 C,C 则再将其发向 D。时间是分隙的,报文所用的传输时间正好是一个时隙,如在时隙 Aloha 中的情况一样。在一个时隙中,节点能够做下列工作之一:(i)发送一个报文(如果它有报文向 D 转发);(ii)接收一个报文(如果正好一个报文要向它发送);(iii)保持静默。如同通常情况那样,如果一个节点听到了两个或更多的节点同时发送,出现冲突,并且重传的报文没有一个能成功收到。你这时能够假定没有比特级的差错,因此如果正好只有一个报文在发送,它将被位于发送方传输半径之内的站点正确收到。

a. 现在假定一个无所不知的控制器(即一个知道在网络中每个节点状态的控制器)能够命令每个节点去做它(无所不知的控制器)希望做的事情,例如发送报文,接收报文,或保持静默。给定这种无所不知的控制器,数据报文能够从 C 到 A 传输的最大速率是什么,假定在任何其他源/目的地对之间没有其他报文?

b. 现在假定 A 向 B 发送报文,并且 D 向 C 发送报文。数据报文能够从 A 到 B 且从 D 到 C 流动的组合最大速率是多少?

c. 现在假定 A 向 B 发送报文且 C 向 D 发送报文。数据报文能够从 A 到 B 且从 C 到 D 流动的组合最大速率是多少?

d. 现在假定无线链路由有线链路代替。在此情况下,重复问题(a)~(c)。

e. 现在假定我们又在无线状态下,对于从源到目的地的每个数据报文,目的地将向源回送一个 ACK 报文(例如,如同在 TCP 中)。对这种情况重复问题(a)~(c)。

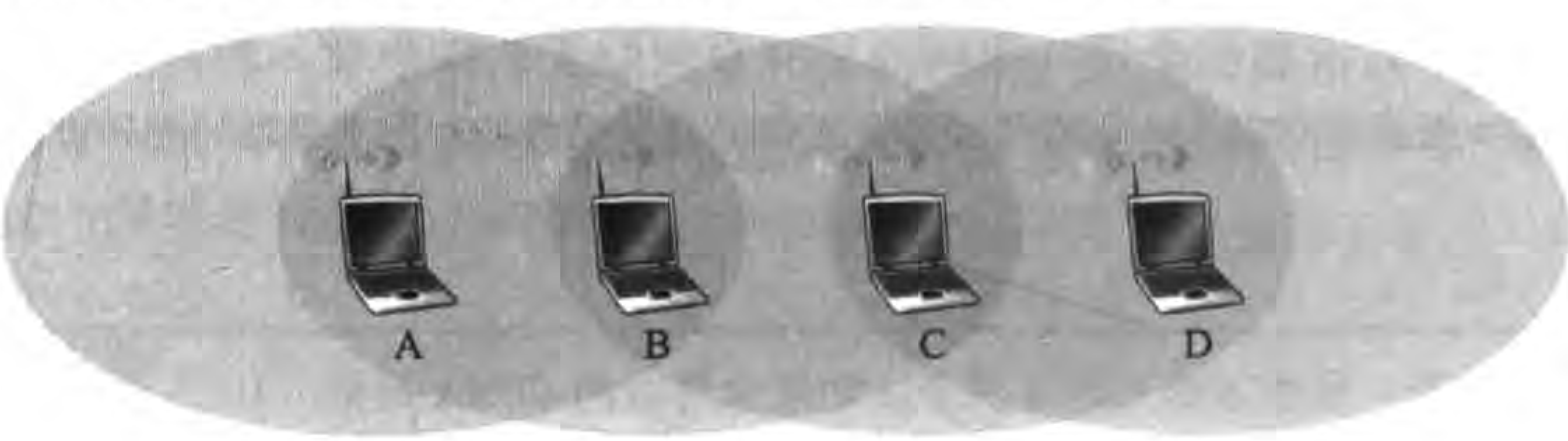


图 7-34 习题 P8 的情形

P9. 描述 802.15.1 蓝牙帧的格式。你必须要阅读某些课外读物来获取这些信息。在帧格式中有哪些因素本质上会限制 802.15.1 网络中主动节点数量为 8 呢?试解释之。

P10. 考虑下列理想化的 LTE 情形。下行信道(参见图 7-21)划分为时隙,使用了 F 个频率。有 4 个节点 A、B、C 和 D 分别以 10Mbps、5Mbps、2.5Mbps 和 1Mbps 速率在下行信道上可到达基站。这些速率假定基本在所有 F 个频率上能够利用所有时隙只向一个站点进行发送。基站具有无限量的数据向每个节点发送,并且在下行子帧中的任何时隙期间使用 F 个频率中的任何之一能够向这 4 个站点之一发送。

a. 假定基站在每个时隙期间能够向它选择的任何节点发送,它能向节点发送的最大速率是多少?你的解决方案公平吗?解释并定义你所指"公平"的含义。

b. 如果有公平要求,即每个站点在每秒期间必须收到等量的数据,在下行子帧期间基站(向所有

节点）的平均传输速率是多少？

c. 假定该公平性准则是在子帧期间任何节点至多能够接收任何其他节点两倍多的数据。在下行子帧期间基站（向所有节点）的平均传输速率是多少？解释你是如何得到答案的。

P11. 在 7.5 节，一种允许移动用户在外部网络间移动过程中保持其 IP 地址不变的建议方案是，让外部网络通告一个到该移动用户高度特定的路由，并使用现有的路由选择基础设施在整个网络中传播这一信息。我们将扩展性作为一种关注因素。假设移动用户从一个网络移动到另一个网络后，新的外部网络通告一个到移动用户的特定路由，旧的外部网络丢弃其路由。考虑路由信息如何在一个距离向量算法中传播（尤其是对于跨越全球的网络间的域间路由选择情况）。

a. 一旦外部网络开始通告其路由，其他路由器能否立刻将数据报路由选择到新的外部网络呢？

b. 不同的路由器有可能认为移动用户位于不同的外部网络中吗？

c. 讨论网络中其他路由器最终知道到达移动用户的路径所用的时间范围。

P12. 假设图 7-23 中通信者是移动的。概述为了将数据报从初始移动用户路由选择到（现在移动的）通信者所需要的额外的网络层基础设施。如图 7-24 中那样，显示在初始移动用户和（现在移动的）通信者之间数据报的结构。

P13. 在移动 IP 中，移动性将对数据报在源和目的地间的端到端时延有怎样的影响？

P14. 考虑 7.7.2 节最后讨论的链的例子。假设一个移动用户访问外部网络 A、B 和 C，当通信者在外部网络 A 中时，它开始一条与移动用户的连接。列出在外部代理之间和外部代理与归属代理之间，当移动用户从网络 A 到网络 B 再到网络 C 的过程中的报文序列。然后，假设未执行链接，并且通信者（以及归属代理）必须被显式地告知移动用户转交地址的改变。列出在第二种情况下需要交换的报文序列。

P15. 考虑在一个具有外部代理的外部网络中的两个移动节点。在移动 IP 中，这两个移动节点是否可能使用相同的转交地址？解释你的答案。

P16. 在我们对 VLR 如何用移动用户当前位置信息更新 HLR 的讨论中，与 VLR 地址对 HLR 相比，提供 MSRN 所具有的优缺点各是什么？

Wireshark 实验

在本书的配套 Web 站点上（http://www.pearsonhighered.com/cs-resources），你将找到有关本章的一个 Wireshark 实验，该实验用于捕获和学习在无线便携机和接入点之间交换的 802.11 帧。

人物专访



Deborah Estrin

DeborahEstrin 是位于纽约城的 Cornell 工学院的计算机科学教授，同时是 Weill Cornell 医学院的公共卫生学教授。她是 Cornell 工学院健康技术中心的创始人和非营利组织创业公司 Open mHealth 的联合奠基人。她从 MIT 获得了计算机科学博士学位（1985 年），并从加州大学伯克利分校获得学士学位（1980 年）。Estrin 的早期研究集中在多播和域间路由选择等网络协议的设计。在 2002 年，Estrin 创建了美国国家自然科学基金资助的科学技术中心 CENS（http://cens.ucla.edu）。CENS 开创了多学科计算机系统研究的新领域，从用于环境监测的传感器网络到用于居民科学的分享感知。她当前致力于移动健康和小数据，促进移动设备和用于健康及生活管理的数字交互的普适性，正如她在 2013 TEDMED 演讲中所描述的。Estrin 教授是美国艺术和科学研究院（2007 年）、国家工程院（2009 年）的当选成员，是 IEEE、ACM 和 AAAS 的会士。她被选为首名 ACM-W 雅典娜讲师（2006 年），获得 Anita Borg 学院的妇女远见创新奖（2007 年），被引入 WITI 名人纪念馆（2008 年），被授予 EPFL（2008 年）和 Uppsala 大学（2011 年）的荣誉博士。

- 请描述您职业生涯中干过的几个最令人激动的项目？

20世纪90年代中期当我在USC和ISI的时候，非常荣幸地与像Steve Deering、Mark Handley和Van Jacobson这样的人物在一起工作，设计多播路由选择协议（特别是PIM）。我试图将多播体系结构设计中的许多经验教训借鉴到生态监视阵列中，这是我首次真正开始全身心地进行应用和多学科的研究。那让我对社会和技术领域中的联合创新感兴趣，它们激发我对近期的研究领域——移动健康的研究兴趣。这些项目中的挑战随问题领域不同而不同，但它们的共同之处是需要睁大我们的眼睛，当我们在设计、部署、制作原型和试用之间重复时关注对问题的定义是否正确。没有一个问题能够借助于模拟或者构造的实验室实验加以分析解决。面对凌乱的问题和环境要保持清晰的体系结构，它们都对我们的能力提出挑战，并且它们都需要广泛的协作。

- 未来在无线网络和移动性方面您预见将会发生什么变化和创新？

在前个版本的专访中曾说过我从来对预测未来不具太多信心，但我的确继续推测，随着智能手机变得越来越强大和因特网基本接入点增多，我们可能看到特色电话（即那些不可编程和仅能用于语音和文本信息的电话）的终结，并且很快在今天这个推测显然已经成真。我也认为我们将看到嵌入式SIM的继续迅速增长，各种设备通过嵌入式SIM经过蜂窝网络能够以低数据率通信。而这种情况已出现，我们看到许多设备和“物联网”，它们使用嵌入式WiFi和其他低功率、短距离以及各种形式连接到本地中心。我并不期待出现大型可穿戴消费市场的时代。但到了本书下个版本出版的时间，我期待着在物联网和其他数字设备数据的促进下，个人应用软件会有极大的增长。

- 您预见网络和因特网未来将往何处发展？

我仍然认为向后看和向前看是有用的。以前我观察到在命名数据和软件定义网络方面的努力将出现成果，产生更可管理、可演化和更丰富的基础设施，并且更一般地表现为推动体系结构的角色向协议栈较高层发展。在因特网初期时，体系结构包括第四层及以下，位于顶端的应用程序更为竖井式/独块式。现在则是数据和分析控制着传输。SDN的采用已经超出了我一直以来的预期，在本书的第7版中包含SDN的内容使我由衷地感到高兴。然而，从协议栈向上看去，我们占优势的应用越来越多地生存在带围墙的花园之中，无论是移动应用还是如脸书那样的大型消费者平台。随着数据科学和大数据技术的发展，由于与其他应用和平台连接的价值，它们可能有助于引导这些应用跳出藩篱。

- 是谁激发了您的职业灵感？

有三个人出现在我的脑海中。第一个人是Dave Clark，他是因特网界的秘方和无名英雄。早期我有幸在他的左右，看到他在IAB的组织规范和因特网管理方法方面所起的作用，成为大致共识和运行编码的引导者。第二个人是Scott Shenker，他的智慧才华、正直和坚持令我印象深刻。我努力却很难像他那样清晰地定义问题和给出解决方案。无论问题大和小，我发电子邮件征求建议，他总是第一个回复的人。第三个人是我的姐姐Judy Estrin，她将创造性和勇气投入到她的职业，将想法和概念带入市场。没有Judy这类人，因特网技术将不会改变我们的生活。

- 您对进入计算机科学和网络领域的学生有什么忠告？

首先，在你的学术工作中构建一个坚实的基础，与你能够得到的任何、每个现实世界的工作经验相权衡。当你寻找一个工作环境时，在你真正关心的问题领域寻找机会，并且参与到你能够从中学习的思维敏捷的团队中。

计算机网络中的安全

早在1.6节我们就描述了某些非常盛行和危险的因特网攻击，包括恶意软件攻击、拒绝服务、嗅探、源伪装以及报文修改和删除。尽管我们已经学习了有关计算机网络的大量知识，但仍然没有考察如何使网络安全，使其免受那些攻击的威胁。在获得了新的计算机网络和因特网协议的专业知识后，我们现在将深入学习安全通信，特别是计算机网络能够防御那些令人厌恶的坏家伙的原理。

我们首先介绍一下Alice和Bob，这俩人要进行通信，并希望该通信过程是“安全”的。由于本书是一本网络教科书，因此Alice和Bob可以是两台需要安全地交换路由选择表的路由器，也可以是希望建立一个安全传输连接的客户和服务器，或者是两个交换安全电子邮件的电子邮件应用程序，所有这些学习案例都是在本章后面我们要考虑的。总之，Alice和Bob是安全领域中两个众所周知的固定设备，也许因为使用Alice和Bob更为有趣，这与命名为“A”的普通实体需要安全地与命名为“B”的普通实体进行通信的作用是一样的。需要安全通信的例子通常包括不正当的情人关系、战时通信和商业事务往来；我们宁愿用第一个例子而不用后两个例子，并使用Alice和Bob作为发送方和接收方，以第一种情况为背景来讨论问题。

我们说过Alice和Bob要进行通信并希望做到“安全”，那么此处的安全其确切含义是什么呢？如我们将看到的那样，安全性（像爱一样）是多姿多彩的东西；也就是说，安全性有许多方面。毫无疑问，Alice和Bob希望他们之间的通信内容对于窃听者是保密的。他们可能也想要确保当他们进行通信时，确实是在和对方通信，还希望如果他们之间的通信被窃听者篡改，他们能够检测到该通信已被篡改破坏。在本章的第一部分，我们将讨论能够加密通信的密码技术，鉴别正在与之通信的对方并确保报文完整性。

在本章的第二部分，我们将研究基本的密码学原则怎样用于生成安全的网络协议。我们再次采用自顶向下方法，从应用层开始，逐层（上面四层）研究安全协议。我们将研究如何加密电子邮件，如何加密一条TCP连接，如何在网络层提供覆盖式安全性，以及如何使无线LAN安全。在本章的第三部分，我们将考虑运行的安全性，这与保护机构网络免受攻击有关。特别是，我们将仔细观察防火墙和入侵检测系统是怎样加强机构网络的安全性的。

8.1 什么是网络安全

我们还是以要进行“安全”通信的情人Alice和Bob为例，开始网络安全的研究。这确切地意味着什么呢？显然，Alice希望即使他们在一个不安全的媒体上进行通信，也只有Bob能够理解她所发送的报文，其中入侵者（入侵者名叫Trudy）能够在该媒体上截获从Alice向Bob传输的报文。Bob也需要确保从Alice接收到的报文确实是由Alice所发送的，并且Alice要确保和她进行通信的人的确就是Bob。Alice和Bob还要确保他们报文的内容在传输过程中没有被篡改。他们首先要确信能够通信（即无人拒绝他们接入通信所需

的资源)。考虑了这些问题后，我们能够指出**安全通信**（secure communication）具有下列所需要的特性。

- **机密性**（confidentiality）。仅有发送方和希望的接收方能够理解传输报文的内容。因为窃听者可以截获报文，这必须要求报文在一定程度上进行**加密**（encrypted），使截取的报文无法被截获者所理解。机密性的这个方面大概就是通常意义上对于术语安全通信的理解。我们将在8.2节中学习数据加密和解密的密码学技术。
- **报文完整性**（message integrity）。Alice和Bob希望确保其通信的内容在传输过程中未被改变——或者恶意篡改或者意外改动。我们在可靠传输和数据链路协议中遇到的检验和技术在扩展后能够用于提供这种报文完整性，我们将在8.3节中研究该主题。
- **端点鉴别**（end-point authentication）。发送方和接收方都应该能证实通信过程所涉及的另一方，以确信通信的另一方确实具有其所声称的身份。人类的面对面通信可以通过视觉识别轻易地解决这个问题。当通信实体在不能看到对方的媒体上交换报文时，鉴别就不是那么简单了。当某用户要访问一个邮箱时，邮件服务器如何证实该用户就是他所声称的那个人呢？我们将在8.4节中学习端点鉴别技术。
- **运行安全性**（operational security）。几乎所有的机构（公司、大学等）今天都有了与公共因特网相连接的网络。这些网络都因此潜在地能够被危及安全。攻击者能够试图在网络主机中安放蠕虫，获取公司秘密，勘察内部网络配置并发起DoS攻击。我们将在8.9节中看到诸如防火墙和入侵检测系统等运行设备正被用于反制对机构网络的攻击。防火墙位于机构网络和公共网络之间，控制接入和来自网络的分组。入侵检测系统执行“深度分组检查”任务，向网络管理员发出有关可疑活动的警告。

明确了我们所指的网络安全的具体含义后，接下来考虑入侵者可能要访问的到底是哪些信息，以及入侵者可能采取哪些行动。图8-1阐述了一种情况。Alice（发送方）想要发送数据给Bob（接收方）。为了安全地交换数据，即在满足机密性、端点鉴别和报文完整性要求的情况下，Alice和Bob交换控制报文和数据报文（以非常类似于TCP发送方和接收方双方交换控制报文和数据报文的方式进行）。通常将这些报文全部或部分加密。如在1.6节所讨论的那样，入侵者能够潜在地执行下列行为：

- **窃听**——监听并记录信道上传输的控制报文和数据报文。
- **修改、插入或删除报文或报文内容。**

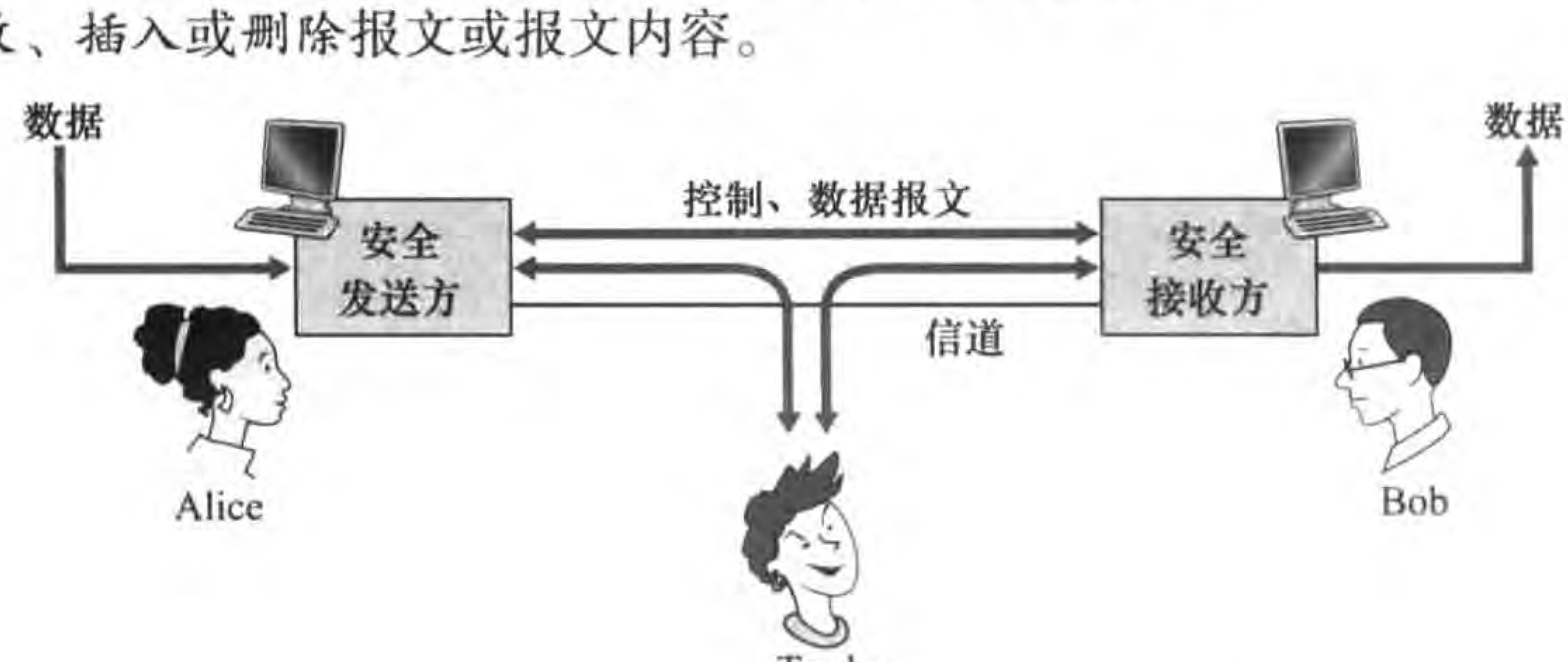


图8-1 发送方、接收方和入侵者（Alice、Bob和Trudy）

如我们将看到的那样，除非采取适当的措施，否则上述能力使入侵者可以用多种方式

发动各种各样的安全攻击：窃听通信内容（可能窃取口令和数据），假冒另一个实体，“劫持”一个正在进行的会话，通过使系统资源过载拒绝合法网络用户的服务请求等等。CERT 协调中心对已报道的攻击进行了总结 [CERT 2016]。

已经知道在因特网中某处的确存在真实的威胁，则 Alice 和 Bob（两个需要安全通信的朋友）在因特网上的对应实体是什么呢？当然，Alice 和 Bob 可以是位于两个端系统的人类用户，例如，真实的 Alice 和真实的 Bob 真的需要交换安全电子邮件。他们也可以参与电子商务事务。例如，真实的 Bob 希望安全地向一台 Web 服务器传输他的信用卡号码，以在线购买商品。类似地，真实的 Alice 要与银行在线交互。需要安全通信的各方自身也可能是网络基础设施的一部分。前面讲过，域名系统（DNS，参见 2.4 节）或交换路由选择信息的路由选择守护程序（参见第 5 章）需要在两方之间安全通信。对于网络管理应用也有相同的情况，第 5 章讨论了该主题。主动干扰 DNS 查找和更新（如在 2.4 节中讨论的那样）、路由选择计算 [RFC 4272] 或网络管理功能 [RFC 3414] 的入侵者能够给因特网造成不可估量的破坏。

建立了上述框架，明确了一些重要定义以及网络安全需求之后，我们将深入学习密码学。应用密码学来提供机密性是不言而喻的，同时我们很快将看到它对于提供端点鉴别、报文完整性也起到了核心作用，这使得密码学成为网络安全的基石。

8.2 密码学的原则

尽管密码学的漫长历史可以追溯到朱利叶斯·凯撒（Julius Caesar）时代，但现代密码技术（包括今天的因特网中正在应用的许多技术）基于过去 30 年所取得的进展。Kahn 的著作《破译者（The Codebreakers）》（[Kahn 1967] 和 Singh 的著作《编码技术：保密的科学——从古埃及到量子密码（The Code Book：The Science of Secrecy from Ancient Egypt to Quantum Cryptography）》[Singh 1999] 回顾了引人入胜的密码学的悠久历史。对密码学的全面讨论需要一本完整的书 [Kaufman 1995；Schneier 1995]，所以我们只能初步了解密码学的基本方面，特别是因为这些东西正在今天的因特网上发挥作用。我们也注意到，尽管本节的重点是密码学在机密性方面的应用，但我们将很快看到密码学技术与鉴别、报文完整性和不可否认性等是紧密相关的。

密码技术使得发送方可以伪装数据，使入侵者不能从截取到的数据中获得任何信息。当然，接收方必须能够从伪装的数据中恢复出初始数据。图 8-2 说明了一些重要的术语。

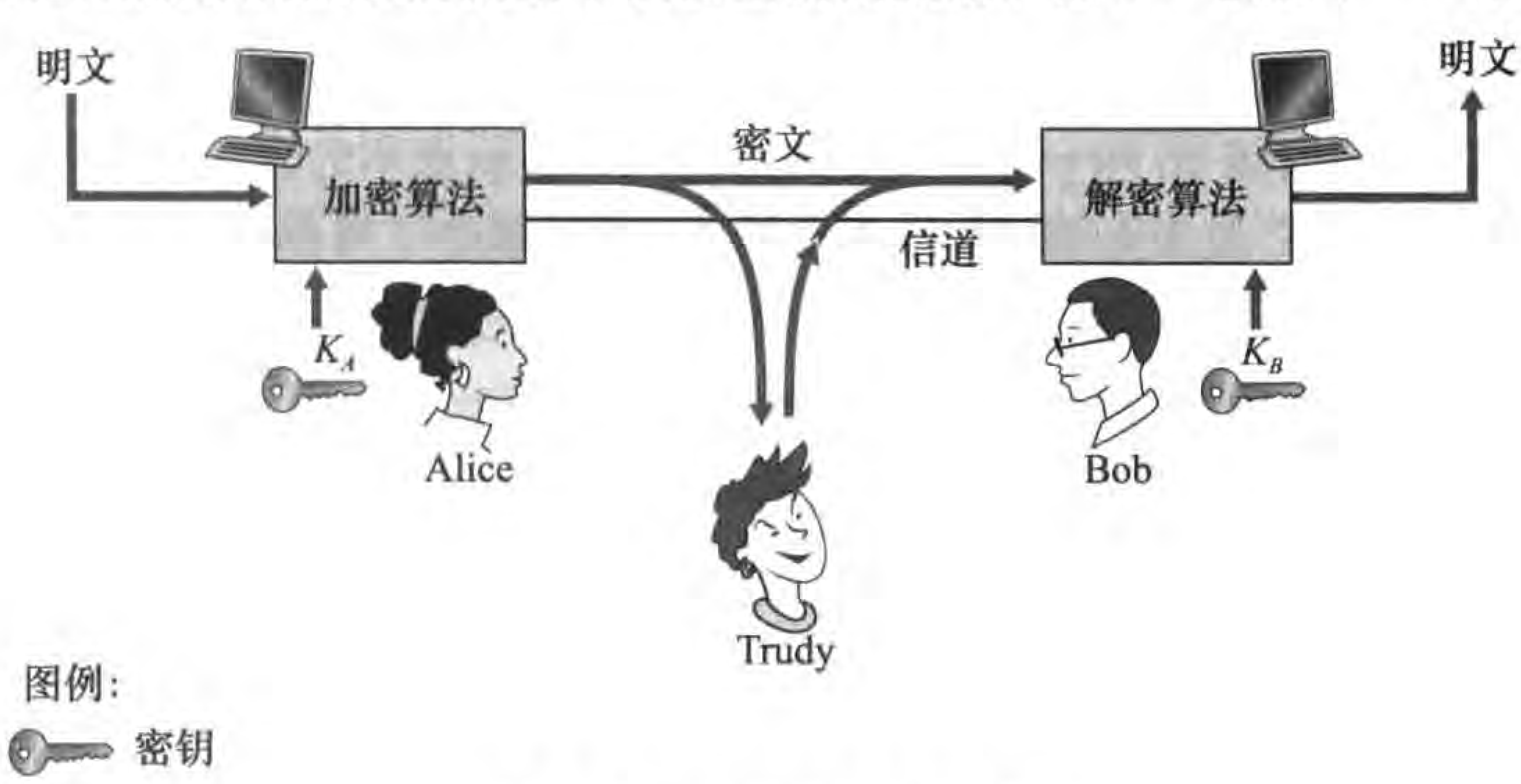


图 8-2　密码学的组成部分

现在假设Alice要向Bob发送一个报文。Alice报文的最初形式（例如，“Bob，I love you. Alice”）被称为**明文**（plaintext，cleartext）。Alice使用**加密算法**（encryption algorithm）加密其明文报文，生成的加密报文被称为**密文**（ciphertext），该密文对任何入侵者看起来是不可懂的。有趣的是在许多现代密码系统中，包括因特网上所使用的那些，加密技术本身是已知的，即公开发行的、标准化的和任何人都可使用的（例如［RFC 1321；RFC 3447；RFC 2420；NIST 2001］），即使对潜在的入侵者也是如此！显然，如果任何人都知道数据编码的方法，则一定有一些秘密信息可以阻止入侵者解密被传输的数据。这些秘密信息就是密钥。

在图8-2中，Alice提供了一个**密钥**（key）K_A，它是一串数字或字符，作为加密算法的输入。加密算法以密钥和明文报文m为输入，生成的密文作为输出。用符号$K_A(m)$表示（使用密钥K_A加密的）明文报文m的密文形式。使用密钥K_A的实际加密算法显然与上下文有关。类似地，Bob将为**解密算法**（decryption algorithm）提供密钥K_B，将密文和Bob的密钥作为输入，输出初始明文。也就是说，如果Bob接收到一个加密的报文$K_A(m)$，他可通过计算$K_B(K_A(m))=m$进行解密。在**对称密钥系统**（symmetric key system）中，Alice和Bob的密钥是相同的并且是秘密的。在**公开密钥系统**（public key system，也称为公钥系统）中，使用一对密钥：一个密钥为Bob和Alice俩人所知（实际上为全世界所知），另一个密钥只有Bob或Alice知道（而不是双方都知道）。在下面两小节中，我们将更为详细地考虑对称密钥系统和公钥系统。

8.2.1　对称密钥密码体制

所有密码算法都涉及用一种东西替换另一种东西的思想，例如，取明文的一部分进行计算，替换适当的密文以生成加密的报文。在分析现代基于密钥的密码系统之前，我们首先学习一下**凯撒密码**（Caesar cipher）找找感觉，这是一种加密数据的方法。这种非常古老而简单的对称密钥算法由Julius Caesar发明。

凯撒密码用于英语文本时，将明文报文中的每个字母用字母表中该字母后第k个字母进行替换（允许回绕，即把字母“a”排在字母“z”之后）。例如，如果$k=3$，则明文中的字母“a”变成密文中的字母“d”；明文中的字母“b”变成密文中的字母“e”，依此类推。因此，k的值就作为密钥。举一个例子，明文报文“bob，i love you. alice”在密文中变成“ere，l oryh brx. dolfh”。尽管密文看起来像乱码，但如果你知道使用了凯撒密码加密，因为密钥值只有25个，所以用不了多久就可以破解它。

凯撒密码的一种改进方法是**单码代替密码**（monoalphabetic cipher），也是使用字母表中的一个字母替换该字母表中的另一个字母。然而，并非按照规则的模式进行替换（例如，明文中的所有字母都用偏移量为k的字母进行替换），只要每个字母都有一个唯一的替换字母，任一字母都可用另一字母替换，反之亦然。图8-3为加密明文的一种可行替换规则。

明文字母:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
密文字母:	m	n	b	v	c	x	z	a	s	d	f	g	h	j	k	l	p	o	i	u	y	t	r	e	w	q

图8-3　单码代替密码

明文报文“bob，i love you. alice”变成“nkn，s gktc wky. mgsbc”。因此，与用凯撒密码情况一样，这看起来像乱码。单码代替密码的性能看来要比凯撒密码的性能好得多，可能的字母配对为26！（10^{26}数量级）种，而不是25种。尝试所有10^{26}种可能配对的蛮力法

要求的工作量太大，不是一种破解加密算法和解密报文的可行方式。但是，通过对明文语言进行统计分析，例如，在典型的英语文本中，由于已知字母“e”和字母“t”出现的频率较高（这些字母出现的频率分别为13%和9%），并且常见的两三个字母的组合通常一起出现（例如，“in”“it”“the”“ion”“ing”等等），这就使得破解该密文变得相对容易。如果入侵者具有某些该报文的可能内容的知识，则破解该密码就会更为容易。例如，如果入侵者 Trudy 是 Bob 的妻子，怀疑 Bob 和 Alice 有暧昧关系，则她猜想“bob”和“alice”这些名字可能会出现在密文中。如果 Trudy 确信这两个名字出现在密文中，并有了上述报文的密文副本，则她能够立即决定这26个字母配对中的7个，比蛮力法少检查 10^9 种可能性。如果 Trudy 的确怀疑 Bob 有不正当的男女关系，她可能也非常期待从该报文中找到某些其他选择的词汇。

当考虑 Trudy 破解 Bob 和 Alice 之间加密方案的难易程度时，可以根据入侵者所拥有的信息分为三种不同的情况。

- *唯密文攻击*。有些情况下，入侵者只能得到截取的密文，也不了解明文报文的内容。我们已经看到，统计分析有助于对加密方案的**唯密文攻击**（ciphertext-only attack）。
- *已知明文攻击*。前面已经看到，如果 Trudy 以某种方式确信在密文报文中会出现“bob”和“alice”，她就可以确定字母 a、l、i、c、e、b 和 o 的（明文，密文）匹配关系。Trudy 也可能会幸运地记录到传输的所有密文，然后在一张纸上找到 Bob 写下的已解密的明文。当入侵者知道（明文，密文）的一些匹配时，我们将其称为对加密方案的**已知明文攻击**（known-plaintext attack）。
- *选择明文攻击*。在**选择明文攻击**（chosen-plaintext attack）中，入侵者能够选择某一明文报文并得到该明文报文对应的密文形式。对于我们前面所说的简单加密算法来说，如果 Trudy 能让 Alice 发送报文“The quick brown fox jumps over the lazy dog”，则 Trudy 就能够完全破解 Alice 和 Bob 所使用的加密方案。但是随后我们将看到，对于更为复杂的加密技术来说，使用选择明文攻击不一定意味着能够攻破该加密机制。

500年前，发明了**多码代替密码**（polyalphabetic encryption），这种技术是对单码代替密码的改进。多码代替密码的基本思想是使用多个单码代替密码，一个单码代替密码用于加密某明文报文中一个特定位置的字母。因此，在某明文报文中不同位置出现的相同字母可能以不同的方式编码。图8-4中显示了多码代替密码机制的一个例子。它使用两个凯撒密码（其中 $k=5$ 和 $k=19$），如图中不同的行所示。我们可以选择使用这两个凯撒密码 C_1 和 C_2，加密时采用以 C_1，C_2，C_2，C_1，C_2 的次序循环的模式，即明文的第一个字母用 C_1 编码，第二和第三个字母用 C_2 编码，第四个字母用 C_1 编码，第五个字母用 C_2 编码，然后循环重复该模式，即第六个字母用 C_1 编码，第七个字母用 C_2 编码，依此类推。这样一来，明文报文“bob, i love you.”加密后成为“ghu, n etox dhz.”。注意到明文报文中的第一个“b”用 C_1 加密为“g”，第二个“b”用 C_2 加密为“u”。在这个例子中，加密和解密“密钥”是两个凯撒密码密钥（$k=5$ 和 $k=19$）和 C_1，C_2，C_2，C_1，C_2 的次序模式的知识。

明文字母	: a b c d e f g h i j k l m n o p q r s t u v w x y z
$C_1(k=5)$	: f g h i j k l m n o p q r s t u v w x y z a b c d e
$C_2(k=19)$	: t u v w x y z a b c d e f g h i j k l m n o p q r s

图8-4 使用两个凯撒密码的多码代替密码

1. 块密码

我们现在回到现代社会中，考察对称密钥加密今天的工作方式。对称加密技术有两种宽泛的类型：**流密码**（stream cipher）和**块密码**（block cipher）。当我们研究无线 LAN 的安全性时，将在 8.7 节中简要地研究流密码。在本节中，我们关注块密码，该密码用在多种因特网协议的加密中，包括 PGP（用于安全电子邮件）、SSL（用于使 TCP 连接更安全）和 IPsec（用于使网络层传输更安全）。

在块密码中，要加密的报文被处理为 k 比特的块。例如，如果 $k=64$，则报文被划分为 64 比特的块，每块被独立加密。为了加密一个块，该密码采用了一对一映射，将 k 比特块的明文映射为 k 比特块的密文。我们来看一个例子。假设 $k=3$，因此块密码将 3 比特输入（明文）映射为 3 比特输出（密文）。表 8-1 给出了一种可能的映射。注意到这是一个一对一的映射，即对每种输入有不同的输出。这种块密码将报文划分成 3 比特的块并根据映射关系进行加密。可以验证，报文 010110001111 被加密成了 101000111001。

表 8-1　一种特定的 3 比特块密码

输入	输出	输入	输出
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

继续这个 3 比特块的例子，注意到上述映射只是许多可能映射中的一种。有多少种可能的映射呢？要回答这个问题，观察到一个映射只不过是所有可能输入的排列。共有 $2^3(=8)$种可能的输入（排列在“输入”栏中）。这 8 种输入能够排列为 $8!=40\ 320$ 种不同方式。因为这些排列的每种都定义了一种映射，共有 40 320 种可能的映射。我们能够将这些映射的每种视为一个密钥，即如果 Alice 和 Bob 都知道该映射（密钥），他们能够加密和解密在他们之间发送的报文。

对这种密码的蛮力攻击即通过使用所有映射来尝试解密密文。仅使用 40 320 种映射（当 $k=3$），这能够在一台桌面 PC 上迅速完成。为了挫败蛮力攻击，块密码通常使用大得多的块，由 64 比特甚至更多比特组成。注意到对于通常的 k 比特块密码，可能映射数量是 $2^k!$，对于即使不大的 k 值（如 $k=64$），这也是一个天文数字。

如刚才所述，尽管全表块密码对于不大的 k 值能够产生健壮的对称密钥加密方案，但不幸的是它们难以实现。对于 $k=64$ 和给定的映射，将要求 Alice 和 Bob 维护一张具有 2^{64} 个输入值的表，这是一个难以实现的任务。此外，如果 Alice 和 Bob 要改变密钥，他们将不得不每人重新生成该表。因此，全表块密码在所有输入和输出之间提供了预先决定的映射（如在上述例子中那样），这简直是不可能实现的事。

取而代之的是，块密码通常使用函数模拟随机排列表。在图 8-5 中显示了当$k=64$时这种函数的一个例子（引自［Kaufman 1995］）。该函数首先将 64 比特块划分为 8 个块，每个块由 8 比特组成。每个 8 比特块由一个“8 比特到 8 比特”表处理，这是个可管理的长度。例如，第一个块由标志为 T_1 的表来处理。接下来，这 8 个输出块被重新装配成一个 64 比特的块。该输出被回馈到 64 比特的输入，开始了第二次循环。经 n 次这样的循环后，该函数提供了一个 64 比特的密文块。这种循环的目的是使得每个输入比特影响最后输出比特的大部分（即使不是全部）。（如果仅使用一次循环，一个给定的输入比特将仅

影响 64 输出比特中的 8 比特。）这种块密码算法的密钥将是 8 张排列表（假定置乱函数是公共已知的）。

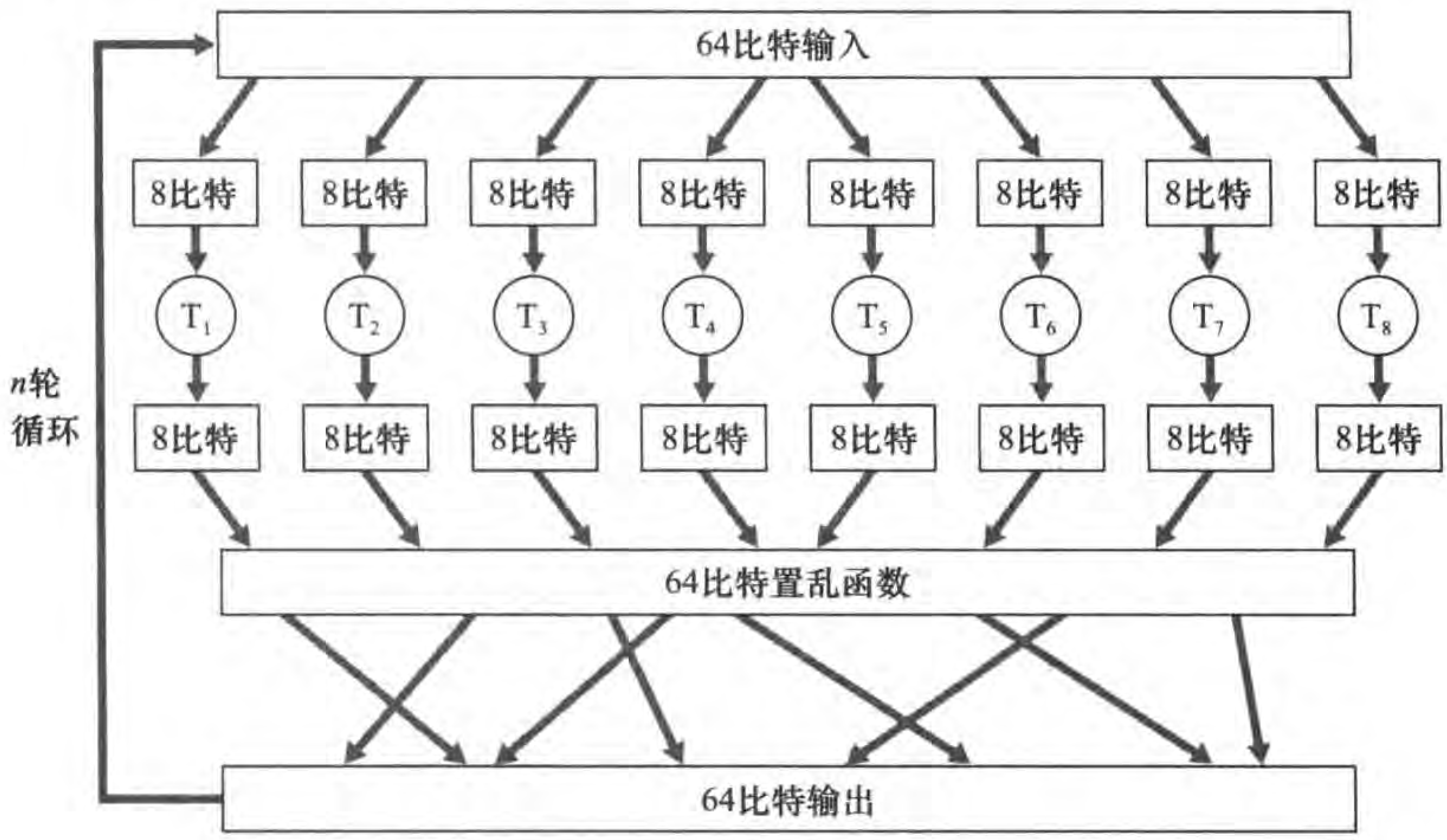


图 8-5 一个块密码的例子

目前有一些流行的块密码，包括 DES（Data Encryption Standard，数据加密标准）、3DES 和 AES（Advanced Encryption Standard，高级加密标准）。这些标准都使用了函数（而不是预先决定的表），连同图 8-5 的线（虽然对每种密码来说更为复杂和具体）。这些算法也都使用了比特串作为密钥。例如，DES 使用了具有 56 比特密钥的 64 比特块。AES 使用 128 比特块，能够使用 128、192 和 256 比特长的密钥进行操作。一个算法的密钥决定了特定“小型表”映射和该算法内部的排列。对这些密码进行蛮力攻击要循环通过所有密钥，用每个密钥应用解密算法。观察到采用长度为 n 的密钥，有 2^n 种可能的密钥。NIST [NIST 2001] 估计，如果用 1 秒破解 56 比特 DES 的计算机（就是说，每秒尝试所有 2^{56} 个密钥）来破解一个 128 比特的 AES 密钥，要用大约 149 万亿年的时间才有可能成功。

2. 密码块链接

在计算机网络应用中，通常需要加密长报文（或长数据流）。如果使用前面描述的块密码，通过直接将报文切割成 k 比特块并独立地加密每块，将出现一个微妙而重要的问题。为了理解这个问题，注意到两个或更多个明文块可能是相同的。例如，两个或更多块中的明文可能是“HTTP/1.1”。对于这些相同的块，块密码当然将产生相同的密文。当攻击者看到相同的密文块时，它可能潜在地猜出其明文，并且通过识别相同的密文块和利用支撑协议结构的知识，甚至能够解密整个报文 [Kaufman 1995]。

为了解决这个问题，可以在密文中混合某些随机性，使得相同的明文块产生不同的密文块。为了解释这个想法，令 $m(i)$ 表示第 i 个明文块，$c(i)$ 表示第 i 个密文块，并且 $a \oplus b$ 表示两个比特串 a 和 b 的异或（XOR）。（前面讲过 $0 \oplus 0 = 1 \oplus 1 = 0$ 和 $0 \oplus 1 = 1 \oplus 0 = 1$，并且两个比特串的异或是逐位进行的。例如 $10101010 \oplus 11110000 = 01011010$。）另外，将具有密钥 S 的块密码加密算法表示为 K_S。其基本思想如下：发送方为第 i 块生成一个随机的 k 比特数 $r(i)$，并且计算 $c(i) = K_S(m(i) \oplus r(i))$。注意到每块选择一个新的 k 比特随机数。则发送方发送 $c(1)$、$r(1)$、$c(2)$、$r(2)$、$c(3)$ 和 $r(3)$ 等等。因为接收方接收到 $c(i)$ 和 $r(i)$，它能够通过计算 $m(i) = K_S(c(i) \oplus r(i))$ 而恢复每个明文块。重要

的是注意到下列事实：尽管 $r(i)$ 是以明文发送的，并且因此能被 Trudy 嗅探到，但她无法获得明文 $m(i)$，因为她不知道密钥 K_S。同时注意到如果两个明文块 $m(i)$ 和 $m(j)$ 是相同的，对应的密文块 $c(i)$ 和 $c(j)$ 将是不同的（只要随机数 $r(i)$ 和 $r(i)$ 不同，这种情况出现的概率将很高）。

举例来说，考虑表 8-1 中的 3 比特块密码。假设明文是 010010010。如果 Alice 直接对此加密，没有包括随机性，得到的密文变为 101101101。如果 Trudy 嗅探到该密文，因为这三个密文块都是相同的，她能够正确地推断出这三个明文块都是相同的。现在假设 Alice 产生了随机块 $r(1)=001$、$r(2)=111$ 和 $r(3)=100$，并且使用了上述技术来生成密文 $c(1)=100$、$c(2)=010$ 和 $c(3)=000$。注意到即使明文块相同，三个密文块也是不同的。Alice 则发送 $c(1)$、$r(1)$、$c(2)$ 和 $r(2)$。读者可证实 Bob 能够使用共享的密钥 K_S 获得初始的明文。

精明的读者将注意到，引入随机性解决了一个问题而产生了另一个问题：Alice 必须传输以前两倍的比特。实际上，对每个加密比特，她现在必须再发送一个随机比特，使需要的带宽加倍。为了有效利用该技术，块密码通常使用了一种称为**密码块链接**（Cipher Block Chaining，CBC）的技术。其基本思想是仅随第一个报文发送一个随机值，然后让发送方和接收方使用计算的编码块代替后继的随机数。具体而言，CBC 运行过程如下：

1）在加密报文（或数据流）之前，发送方生成一个随机的 k 比特串，称为**初始向量**（Initialization Vector，IV）。将该初始向量表示为 $c(0)$。发送方以明文方式将 IV 发送给接收方。

2）对第一个块，发送方计算 $m(1)\oplus c(0)$，即计算第一块明文与 IV 的异或。然后通过块密码算法运行得到的结果以得到对应的密文块，即 $c(1)=K_S(m(1)\oplus c(0))$。发送方向接收方发送加密块 $c(1)$。

3）对于第 i 个块，发送方根据 $c(i)=K_S(m(i)\oplus c(i-1))$ 生成第 i 个密文块。

我们现在来考察这种方法的某些后果。首先，接收方将仍能够恢复初始报文。毫无疑问，当接收方接收到 $c(i)$ 时，它用 K_S 解密之以获得 $s(i)=m(i)\oplus c(i-1)$；因为接收方已经知道 $c(i-1)$，则从 $m(i)=s(i)\oplus c(i-1)$ 获得明文块。第二，即使两个明文块是相同的，相应的密文块也（几乎）总是不同的。第三，虽然发送方以明文发送 IV，入侵者将仍不能解密密文块，因为该入侵者不知道秘密密钥 S。最后，发送方仅发送一个最前面的块（即 IV），因此对（由数百块组成的）长报文而言增加的带宽用量微不足道。

举例来说，对表 8-1 中的 3 比特块密码，明文为 010010010 和 $IV=c(0)=001$，我们现在来确定其密文。发送方首先使用 IV 来计算 $c(1)=K_S(m(1)\oplus c(0))=100$。发送方然后计算 $c(2)=K_S(m(2)\oplus c(1))=K_S(010\oplus 100)=000$，并且 $c(3)=K_S(m(3)\oplus c(2))=K_S(010\oplus 000)=101$。读者可证实接收方若知道了 IV 和 K_S，将能够恢复初始的明文。

当设计安全网络协议时，CBC 有一种重要的后果：需要在协议中提供一种机制，以从发送方向接收方分发 IV。在本章稍后我们将看到几个协议是如何这样做的。

8.2.2　公开密钥加密

从凯撒密码时代直到 20 世纪 70 年代的两千多年以来，加密通信都需要通信双方共享一个共同秘密，即用于加密和解密的对称密钥。这种方法的一个困难是两方必须就共享密钥达成一致，但是这样做的前提是需要通信（可假定是安全的）！可能是双方首先会面，

人为协商确定密钥（例如，凯撒的两个百夫长在罗马浴室碰头），此后才能进行加密通信。但是，在网络世界中，通信各方之间可能从未见过面，也不会在网络以外的任何地方交谈。此时通信双方能够在没有预先商定的共享密钥的条件下进行加密通信吗？1976 年，Diffie 和 Hellman［Diffie 1976］论证了一个解决这个问题的算法（现在称为 Diffie-Hellman 密钥交换），这是个完全不同、极为优雅的安全通信算法，开创了如今的公开密钥密码系统的发展之路。我们很快就会看到公开密钥密码系统也有许多很好的特性，使得它不仅可以用于加密，还可以用于鉴别和数字签名。有趣的是，最近发现 20 世纪 70 年代早期由英国通信电子安全团体的研究人员独立研究的一系列秘密报告中的思想，与［Diffie 1976］和［RSA 1978］中的思想类似［Ellis 1987］。事实常常如此，伟大的想法通常会在许多地方独立地闪现；幸运的是，公钥的进展不仅秘密地发生，而且也在公众视野中发生。

公开密钥密码的使用在概念上相当简单。假设 Alice 要和 Bob 通信。如图 8-6 所示，这时 Alice 和 Bob 并未共享一个密钥（如同在对称密钥系统情况下），而 Bob（Alice 报文的接收方）则有两个密钥，一个是世界上任何人（包括入侵者 Trudy）都可得到的**公钥**（public key），另一个是只有 Bob 知道的**私钥**（private key）。我们使用符号 K_B^+ 和 K_B^- 来分别表示 Bob 的公钥和私钥。为了与 Bob 通信，Alice 首先取得 Bob 的公钥，然后用这个公钥和一个众所周知的（例如，已标准化的）加密算法，加密她要传递给 Bob 的报文 m；即 Alice 计算 $K_B^+(m)$。Bob 接收到 Alice 的加密报文后，用其私钥和一个众所周知的（例如，已标准化的）解密算法解密 Alice 的加密报文，即 Bob 计算 $K_B^-(K_B^+(m))$。后面我们将看到，存在着可以选择公钥和私钥的加密/解密算法和技术，使得 $K_B^-(K_B^+(m))=m$；也就是说，用 Bob 的公钥 K_B^+ 加密报文 m（得到 $K_B^+(m)$），然后再用 Bob 的私钥 K_B^- 解密报文的密文形式（就是计算 $K_B^-(K_B^+(m))$）就能得到最初的明文 m。这是个不寻常的结果！用这种办法，Alice 可以使用 Bob 公开可用的密钥给 Bob 发送机密信息，而他们任一方都无须分发任何密钥！我们很快能够看到，公钥和私钥加密相互交换同样能够得到不寻常的结果，即 $K_B^-(K_B^+(m))=K_B^+(K_B^-(m))=m$。

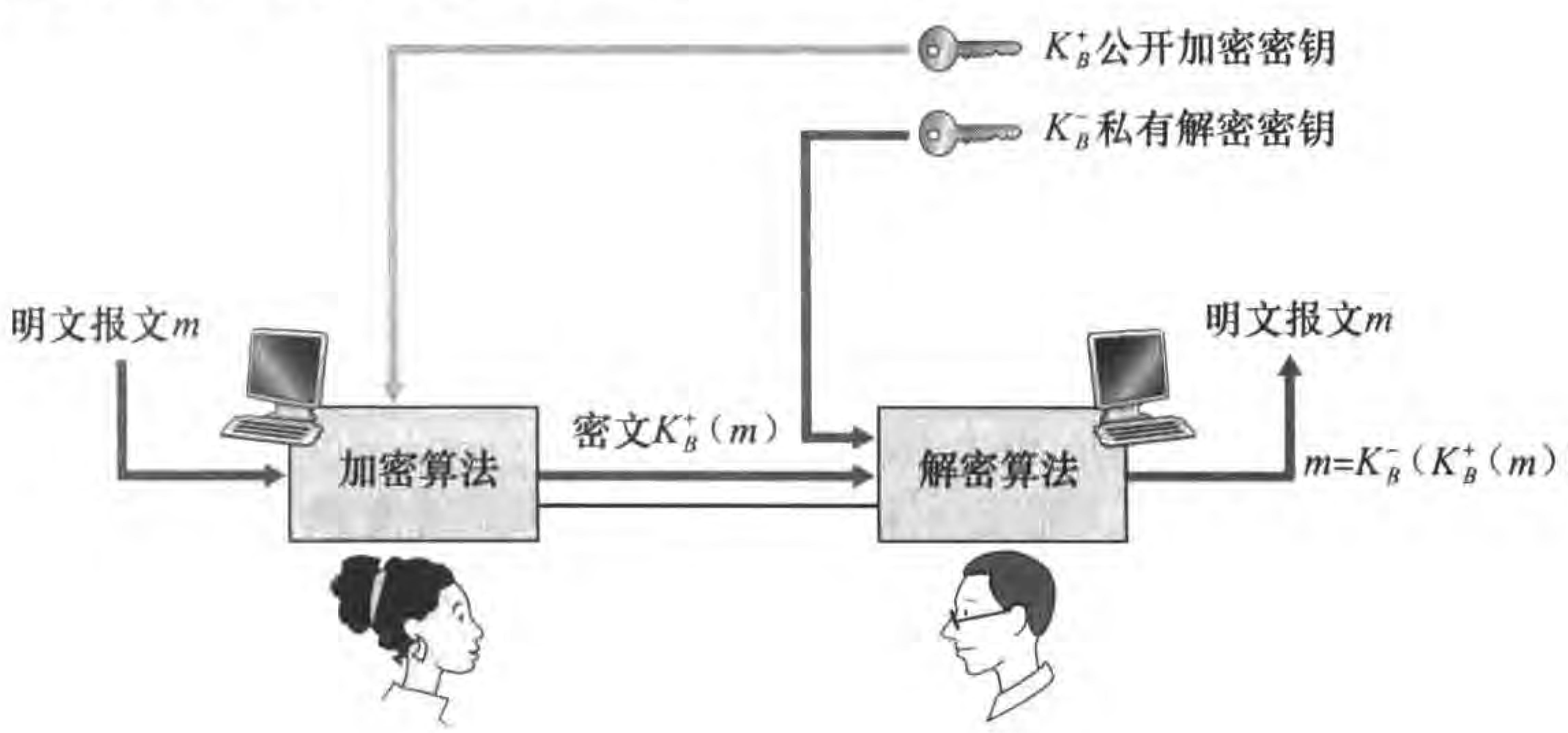


图 8-6 公开密钥密码

因此公开密钥密码体制的使用在概念上是简单的。但是有两点必须要注意。首先应关注的是，尽管入侵者截取到 Alice 的加密报文时看到的只是乱码，但是入侵者知道公钥（显然 Bob 的公钥是全世界都可以使用的）和 Alice 加密所用的算法。Trudy 可以据此发起选择明文攻击，使用已知的标准加密算法和 Bob 的公开可用的加密密钥对她所选择的任意报文编码！例如，Trudy 可以尝试对她怀疑 Alice 可能发送的全部报文或部分报文编码。很

明显，要使公开密钥密码能工作，密钥选择和加密/解密算法必须保证任意入侵者都不能（至少要困难得几乎不可能）确定出 Bob 的私钥或者以某种方式解密或猜出 Alice 发给 Bob 的报文。第二个值得关注的问题是，既然 Bob 的加密密钥是公开的，任何人（包括 Alice 和其他声称自己是 Alice 的人）都可能向 Bob 发送一个已加密的报文。在单一共享密钥情况下，发送方知道共享秘密密钥的事实就已经向接收方隐含地证实了发送方的身份。然而在公钥体制中，这点就行不通了，因为任何一个人都可向 Bob 发送使用 Bob 的公开可用密钥加密的报文。这就需要用数字签名把发送方和报文绑定起来，数字签名是我们在 8.3 节中讨论的主题。

1. RSA

尽管可能有许多算法处理这些关注的问题，但 **RSA 算法**（RSA algorithm，取算法创立人 Ron Rivest、Adi Shamir 和 Leonard Adleman 的首字母命名）几乎已经成了公开密钥密码的代名词。我们首先来理解 RSA 是如何工作的，然后再考察 RSA 的工作原理。

RSA 广泛地使用了模 n 算术的算术运算。故我们简要地回顾一下模算术。前面讲过 $x \bmod n$ 只是表示被 n 除时 x 的余数；因此如 $19 \bmod 5 = 4$。在模算术中，人们执行通常的加法、乘法和指数运算。然而，每个运算的结果由整数余数代替，该余数是被 n 除后留下的数。对于模算术的加法和乘法可由下列便于施用的事实所简化：

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$$
$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$
$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

从第三个事实推出 $(a \bmod n)^d \bmod n = a^d \bmod n$，我们很快将会发现这个恒等式是非常有用的。

现在假设 Alice 要向 Bob 发送一个 RSA 加密的报文，如图 8-6 所示。在我们的讨论中，心中永远要记住一个报文只不过是一种比特模式，并且所有比特模式能唯一地被一个整数（连同该比特模式的长度）表示。例如，假设一个报文是比特模式 1001；这个报文能由十进制整数 9 来表示。所以，当用 RSA 加密一个报文时，等价于加密表示该报文的这个唯一的整数。

RSA 有两个互相关联的部分：

- 公钥和私钥的选择。
- 加密和解密算法。

为了生成 RSA 的公钥和私钥，Bob 执行如下步骤：

1）选择两个大素数 p 和 q。那么 p 和 q 应该多大呢？该值越大，破解 RSA 越困难，而执行加密和解密所用的时间也越长。RSA 实验室推荐，公司使用时，p 和 q 的乘积为 1024 比特的数量级。对于选择大素数的方法的讨论，参见［Caldwell 2012］。

2）计算 $n = pq$ 和 $z = (p-1)(q-1)$。

3）选择小于 n 的一个数 e，且使 e 和 z 没有（非 1 的）公因数。（这时称 e 与 z 互素。）使用字母 e 表示是因为这个值将被用于加密。

4）求一个数 d，使得 $ed-1$ 可以被 z 整除（就是说，没有余数）。使用字母 d 表示是因为这个值将用于解密。换句话说，给定 e，我们选择 d，使得

$$ed \bmod z = 1$$

5）Bob 使外界可用的公钥 K_B^+ 是一对数 (n, e)，其私钥 K_B^- 是一对数 (n, d)。

Alice 执行的加密和 Bob 进行的解密过程如下：

- 假设 Alice 要给 Bob 发送一个由整数 m 表示的比特组合，且 $m<n$。为了进行编码，Alice 执行指数运算 m^e，然后计算 m^e 被 n 除的整数余数。换言之，Alice 的明文报文 m 的加密值 c 就是：

$$c = m^e \bmod n$$

对应于这个密文 c 的比特模式发送给 Bob。

- 为了对收到的密文报文 c 解密，Bob 计算：

$$m = c^d \bmod n$$

这要求使用他的私钥 (n, d)。

举一个简单的 RSA 例子，假设 Bob 选择 $p=5$ 和 $q=7$。（坦率地讲，这样小的值无法保证安全。）则 $n=35$ 和 $z=24$。因为 5 和 24 没有公因数，所以 Bob 选择 $e=5$。最后，因为 $5\times29-1$（即 $ed-1$）可以被 24 整除，所以 Bob 选择 $d=29$。Bob 公开了 $n=35$ 和 $e=5$ 这两个值，并秘密保存了 $d=29$。观察公开的这两个值，假定 Alice 要发送字母 "l" "o" "v" 和 "e" 给 Bob。用 1 ~ 26 之间的每个数表示一个字母，其中 1 表示 "a"，…，26 表示 "z"，Alice 和 Bob 分别执行如表 8-2 和表 8-3 所示的加密和解密运算。注意到在这个例子中，我们认为每四个字母作为一个不同报文。一个更为真实的例子是把这四个字母转换成它们的 8 比特 ASCII 表示形式，然后加密与得到的 32 比特的比特模式对应的整数。（这样一个真实的例子产生了一些长得难以在教科书中打印出来的数！）

表 8-2　Alice 的 RSA 加密，$e=5$，$n=35$

明文字母	m：数字表示	m^e	密文 $c=m^e \bmod n$
l	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

表 8-3　Bob 的 RSA 解密，$d=29$，$n=35$

密文 c	c^d	$m=c^d \bmod n$	明文字母
17	4819685721067509150915091411825223071697	12	l
15	127834039488589391112327575683593375	15	o
22	851643319086537701956194499721106030592	22	v
10	100000000000000000000000000000	5	e

假定表 8-2 和表 8-3 中的简单示例已经产生了某些极大的数，并且假定我们前面看到 p 和 q 都是数百比特长的数，这些都是实际使用 RSA 时必须要牢记的。如何选择大素数？如何选择 e 和 d？如何对大数进行指数运算？对这些重要问题的详细讨论超出了本书的范围，详情请参见 [Kaufman 1995] 以及其中的参考文献。

2. 会话密钥

这里我们注意到，RSA 所要求的指数运算是相当耗费时间的过程。形成对比的是，DES 用软件实现要比 RSA 快 100 倍，用硬件实现则要快 1000 ~ 10 000 倍 [RSA Fast 2012]。所以，在实际应用中，RSA 通常与对称密钥密码结合起来使用。例如，如果 Alice 要向 Bob 发送大量的加密数据，她可以用下述方式来做。首先，Alice 选择一个用于加密数据本身的密钥，这个密钥有时称为**会话密钥**（session key），该会话密钥表示为 K_S。

Alice 必须把这个会话密钥告知 Bob，因为这是他们在对称密钥密码（如 DES 或 AES）中所使用的共享对称密钥。Alice 可以使用 Bob 的 RSA 公钥来加密该会话密钥，即计算 $c=(K_S)^e \bmod n$。Bob 收到了该 RSA 加密的会话密钥 c 后，解密得到会话密钥 K_S。Bob 此时已经知道 Alice 将要用于加密数据传输的会话密钥了。

3. RSA 的工作原理

RSA 加密/解密看起来相当神奇。为什么那样应用加密算法，然后再运行解密算法，就能恢复出初始报文呢？要理解 RSA 的工作原理，我们仍将记 $n=pq$，其中 p 和 q 是 RSA 算法中的大素数。

前面讲过，在 RSA 加密过程中，一个报文 m（唯一地表示为整数）使用模 n 算术做 e 次幂运算，即

$$c = m^e \bmod n$$

解密则先对该值执行 d 次幂运算，再做模 n 运算。因此先加密再解密的结果是 $(m^e \bmod n)^d \bmod n$。下面我们来看关于这个量能够得到什么。正如前面提到的，模算术的一个重要性质是对于任意值 a、n 和 d 都有 $(a \bmod n)^d \bmod n = a^d \bmod n$。因此，在这个性质中使用 $a=m^e$，则有

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

因此剩下证明 $m^{ed} \bmod n = m$。尽管我们正试图揭开 RSA 工作原理的神秘面纱，但为了做到这一点，我们还需要用到数论中一个相当神奇的结果。具体而言，就是要用到数论中这样的结论：如果 p 和 q 是素数，且有 $n=pq$ 和 $z=(p-1)(q-1)$，则 $x^y \bmod n$ 与 $x^{(y \bmod z)} \bmod n$ 是等同的 [Kaufman 1995]。应用这个结论，对于 $x=m$ 和 $y=ed$，可得

$$m^{ed} \bmod n = m^{(ed \bmod z)} \bmod n$$

但是要记住，我们是这样选择 e 和 d 的，即 $ed \bmod z = 1$。这告诉我们

$$m^{ed} \bmod n = m^1 \bmod n = m$$

这正是我们希望得到的结果！先对 m 做 e 次幂运算（加密）再做 d 次幂运算（解密），然后做模 n 的算术运算（原文中没有这句，译者认为有必要补上。——译者注），就可得到初始的 m。甚至更为奇妙之处是这样一个事实，如果我们先对 m 做 d 次幂运算（加密）再做 e 次幂运算，即颠倒加密和解密的次序，先执行解密操作再执行加密操作，也能得到初始值 m。这个奇妙的结果完全遵循下列模算术：

$$(m^d \bmod n)^e \bmod n = m^{de} \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$$

RSA 的安全性依赖于这样的事实：目前没有已知的算法可以快速进行一个数的因数分解，这种情况下公开值 n 无法快速分解成素数 p 和 q。如果已知 p 和 q，则给定公开值 e，就很容易计算出秘密密钥 d。另一方面，不确定是否存在因数分解一个数的快速算法，从这种意义上来说，RSA 的安全性也不是确保的。

另一种流行的公钥加密算法是 Diffie-Hellman，我们将在课后习题中简要探讨它。Diffie-Hellman 并不像 RSA 那样多功能，即它不能用于加密任意长度的报文；然而，它能够用来创建一个对称的会话密钥，该密钥再被用于加密长报文。

8.3 报文完整性和数字签名

在前面一节中我们看到了能够使用加密为两个通信实体提供机密性。在本节中我们转向提供**报文完整性**（message integrity）这个同等重要的主题。报文完整性也称为报文鉴

别。连同报文完整性，在本节中我们将讨论两个相关的主题：数字签名和端点鉴别。

我们再次使用 Alice 和 Bob 来定义报文完整性问题。假定 Bob 接收到一个报文（这可能已经加密或可能是明文），并且他认为这个报文是由 Alice 发送的。为了鉴别这个报文，Bob 需要证实：

1）该报文的确源自 Alice。

2）该报文在到 Bob 的途中没有被篡改。

我们将在 8.4～8.7 节中看到，报文完整性这个问题在所有安全网络协议中都是至关重要的。

举一个特定的例子，考虑一个使用链路状态路由选择算法（例如 OSPF）的计算机网络，在该网络中决定每对路由器之间的路由（参见第 5 章）。在一个链路状态算法中，每台路由器需要向该网络中的所有其他路由器广播一个链路状态报文。路由器的链路状态报文包括直接相连邻居的列表以及到这些邻居的直接费用。一旦某台路由器从其他所有路由器收到了链路状态报文，它能够生成该网络的全图，运行它的最小费用路由选择算法并配置它的转发表。对路由选择算法的一个相对容易的攻击是，Trudy 分发具有不正确状态信息的虚假链路状态报文。因此产生了报文完整性的需求：当路由器 B 收到来自路由器 A 的链路状态报文时，路由器 B 应当证实路由器 A 实际生成了该报文，并且进一步证实在传输过程中该报文没有被篡改。

在本节中，我们描述一种由许多安全网络协议所使用的流行报文完整性技术。但在做此事之前，我们需要涉及密码学中的另一个重要主题，即密码散列函数。

8.3.1 密码散列函数

如图 8-7 所示，散列函数以 m 为输入，并计算得到一个称为散列的固定长度的字符串 $H(m)$。因特网检验和（第 3 章）和 CRC（第 6 章）都满足这个定义。**密码散列函数**（cryptographic hash function）要求具有下列附加的性质：

- 找到任意两个不同的报文 x 和 y 使得 $H(x)=H(y)$，在计算上是不可能的。

不严格地说，这种性质就意味着入侵者在计算上不可能用其他报文替换由散列函数保护的报文。这就是说，如果（m，$H(m)$）是报文和由发送方生成的报文散列的话，则入侵者不可能伪造另一个报文 y 的内容，使得该报文具有与原报文相同的散列值。

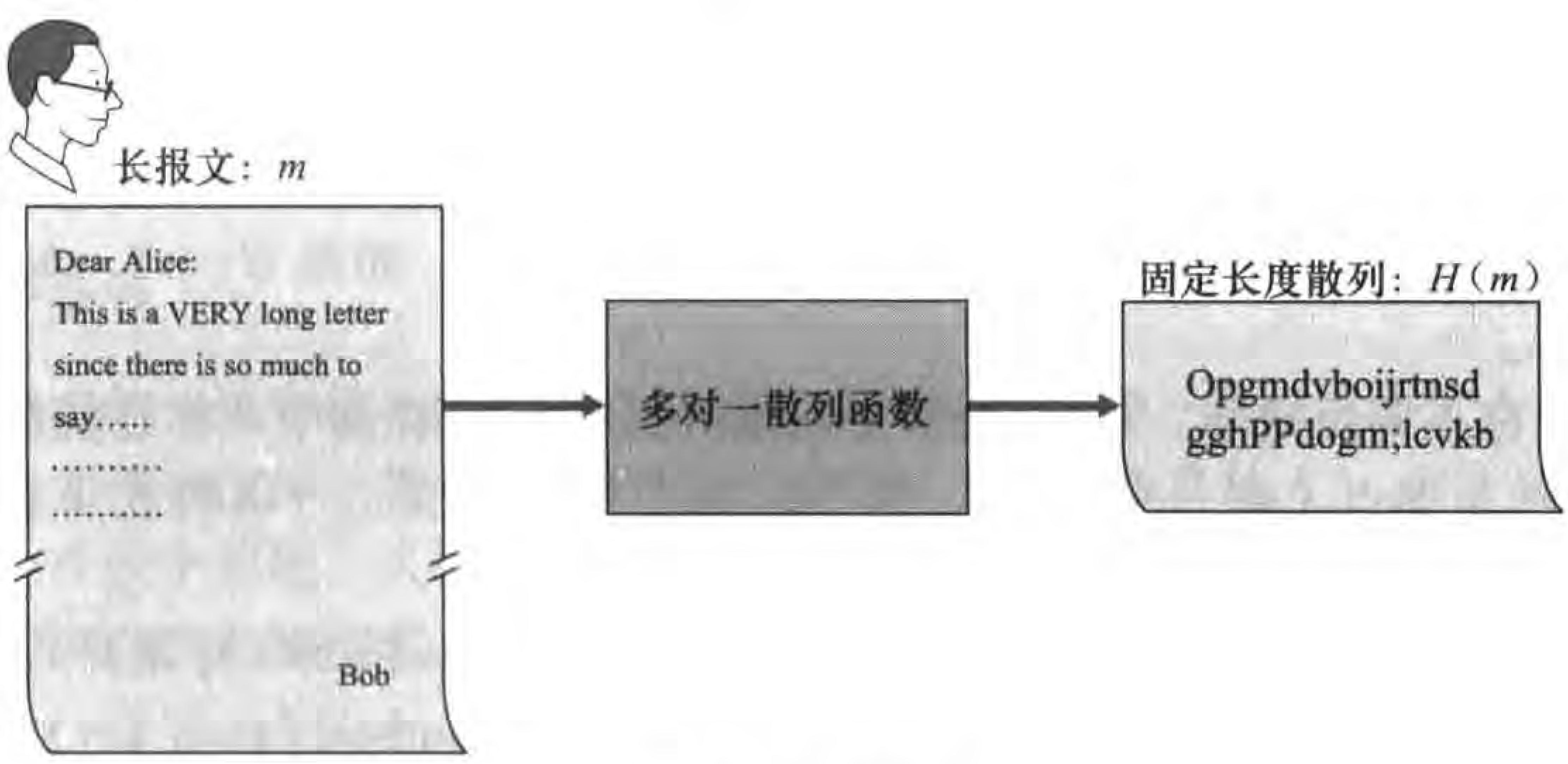


图 8-7 散列函数

我们来证实一个简单的检验和（如因特网检验和）只能算作劣质的密码散列函数。不

像在因特网检验和中执行反码运算那样，我们把每个字符看作一个字节，并把这些字节加到一起，一次用4字节的块来进行计算。假定Bob欠Alice 100.99美元并且向Alice发送一张借据，这个借据包含文本字符串“IOU100.99BOB”。这些字符的ASCII表示（以十六进制形式）为49，4F，55，31，30，30，2E，39，39，42，4F，42。

图8-8上半部分显示了这个报文的4字节检验和是B2 C1 D2 AC。图8-8下半部分显示了一条稍微不同的报文（但是Bob要付的钱却多了许多）。报文“IOU100.99BOB”和“IOU900.19BOB”有相同的检验和。因此，这种简单的检验和算法违反了上述要求。给定初始数据，很容易找到有相同检验和的另一组数据。很明显，为了安全起见，我们需要比检验和更为强有力的散列函数。

Ron Rivest［RFC 1321］的MD5散列算法如今正在广泛使用。这个算法通过4步过程计算得到128比特的散列。这4步过程由下列步骤组成：①填充——先填1，然后填足够多的0，直到报文长度满足一定的条件；②添加——在填充前添加一个用64比特表示的报文长度；③初始化累加器；④循环——在最后的循环步骤中，对报文的16字块进行4轮处理。MD5的描述（包括一个C源代码实现）可参见［RFC 1321］。

报文	ASCII表示				
I O U 1	49	4F	55	31	
0 0 . 9	30	30	2E	39	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	检验和

报文	ASCII表示				
I O U 9	49	4F	55	39	
0 0 . 1	30	30	2E	31	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	检验和

图8-8　初始报文和欺诈报文具有相同的检验和

目前正使用的第二个主要散列算法是安全散列算法SHA-1（Security Hash Algorithm）［FIPS 1995］。这个算法的原理类似于MD4［RFC 1320］设计中所使用的原理，而MD4是MD5的前身。SHA-1是美国联邦政府的标准，任何联邦政府的应用程序如果需要使用密码散列算法的话，都要求使用SHA-1。SHA-1生成一个160比特的报文摘要。较长的输出长度可使SHA-1更安全。

8.3.2　报文鉴别码

我们现在再回到报文完整性的问题。既然我们理解了散列函数，就先来看一下将如何执行报文完整性：

1）Alice生成报文 m 并计算散列 $H(m)$（例如使用SHA-1）。

2）然后Alice将 $H(m)$ 附加到报文 m 上，生成一个扩展报文（m，$H(m)$），并将该扩展报文发给Bob。

3）Bob接收到一个扩展报文（m，h）并计算 $H(m)$。如果 $H(m)=h$，Bob得到结论：一切正常。

这种方法存在明显缺陷。Trudy能够生成虚假报文 m'，在其中声称她就是Alice，计算 $H(m')$ 并发送给Bob（m'，$H(m')$）。当Bob接收到该报文，一切将在步骤3中核对通过，并且Bob无法猜出这种不轨的行为。

为了执行报文完整性，除了使用密码散列函数外，Alice和Bob将需要共享秘密 s。这个共享的秘密只不过是一个比特串，它被称为**鉴别密钥**（authentication key）。使用这个共享秘密，报文完整性能够执行如下：

1）Alice生成报文 m，用 s 级联 m 以生成 $m+s$，并计算散列 $H(m+s)$（例如使用SHA-1）。$H(m+s)$ 被称为**报文鉴别码**（Message Authentication Code，MAC）。

2）然后 Alice 将 MAC 附加到报文 m 上，生成扩展报文（m，$H(m+s)$），并将该扩展报文发送给 Bob。

3）Bob 接收到一个扩展报文（m，h），由于知道 s，计算出报文鉴别码 $H(m+s)$。如果 $H(m+s)=h$，Bob 得到结论：一切正常。

图 8-9 中总结了上述过程。读者应当注意到这里的 MAC（表示“报文鉴别码”）与用于数据链路层中的 MAC（表示“媒体访问控制”）是不一样的！

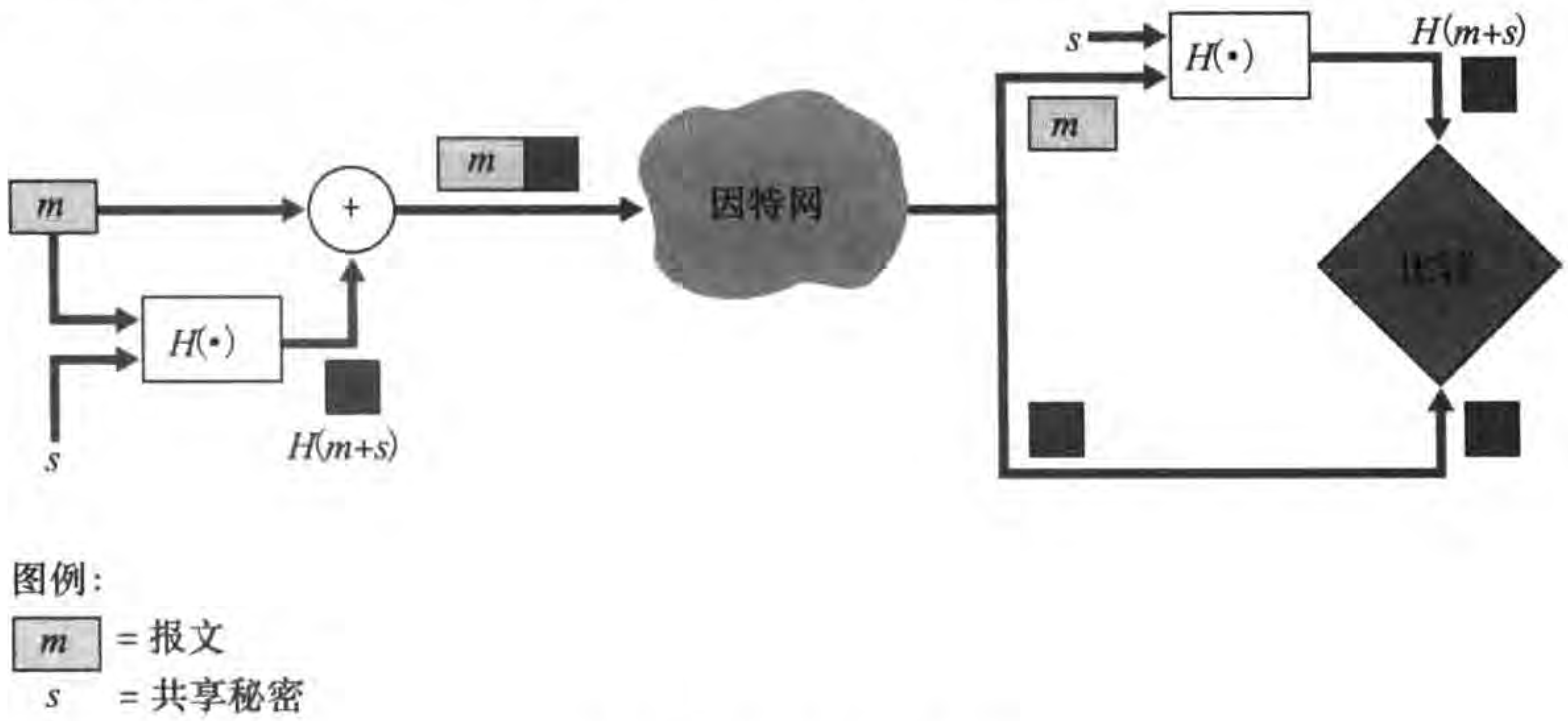


图 8-9　报文鉴别码

MAC 的一个优良特点是它不要求一种加密算法。的确，在许多应用中，包括前面讨论的链路状态路由选择算法，通信实体仅关心报文完整性，并不关心报文机密性。使用 MAC，实体能够鉴别它们相互发送的报文，而不必在完整性过程中综合进复杂的加密过程。

如你所猜测，多年来已经提出了若干种对 MAC 的不同标准。目前最为流行的标准是 HMAC，它能够与 MD5 或 SHA-1 一道使用。HMAC 实际上通过散列函数运行数据和鉴别密钥两次［Kaufman 1995；RFC 2104］。

这里还遗留下一个重要问题。怎样向通信实体分发这个共享的鉴别密钥呢？例如，在链路状态路由选择算法中，在某种程度上需要向自治系统中的每台路由器分发该秘密鉴别密钥。（注意到所有路由器都能够使用相同的鉴别密钥。）一名网络管理员能够通过物理上访问每台路由器来实际完成这项工作。或者，如果这名网络管理员不够勤快，并且每台路由器都有它自己的公钥，那么该网络管理员能够用路由器的公钥加密鉴别密钥并分发给任何一台路由器，从而通过网络向路由器发送加密的密钥。

8.3.3 数字签名

回想在过去的一周中你在纸上已经签过多少次你的名字。你可能经常会在支票、信用卡收据、法律文件和信件上签名。你的签名证明你（而不是其他人）承认和/或同意这些文件的内容。在数字领域，人们通常需要指出一个文件的所有者或创作者，或者表明某人认可一个文件内容。**数字签名**（digital signature）就是一种在数字领域实现这些目标的密码技术。

正如手工签字一样，数字签名也应当以可鉴别的、不可伪造的方式进行。这就是说，必须能够证明由某个人在一个文件上的签名确实是由该人签署的（该签名必须是可证实的），且只有那个人能够签署那个文件（该签名无法伪造）。

我们现在来考虑怎样设计一个数字签名方案。当 Bob 签署一个报文时，可以观察到 Bob 必须将某些对他独特的东西放置在该报文上。Bob 可以考虑附加一个 MAC 用作签名，其中 MAC 是由他的密钥（对他是独特的）作用到该报文上而生成的，然后得到该散列值。而 Alice 为了验证该签名，她必须也具有该密钥的副本，在这种情况下该密钥对 Bob 将不是唯一的。因此，此时 MAC 是无法胜任这项工作的。

前面讲过使用公钥密码，Bob 具有公钥和私钥，这两种密钥对 Bob 均为独特的。因此，公钥密钥是一种提供数字签名的优秀候选者。我们现在来研究一下这是怎样完成的。

假设 Bob 要以数字方式签署一个文档 m。我们能够想象这个文档是 Bob 打算签署并发送的一个文件或一个报文。如图 8-10 所示，为了签署这个文档，Bob 直接使用他的私钥 K_B^- 计算 $K_B^-(m)$。乍一看，会感觉很奇怪，Bob 怎么会用他的私钥（在 8.2 节中，我们用私钥解密用其公钥加密的报文）签署文档！但是回想加密和解密都只不过是数学运算（RSA 中所做的 e 或 d 指数幂运算；参见 8.2 节），并且 Bob 的目的不是弄乱或掩盖文档的内容，而只是以可鉴别、不可伪造的方式签署这个文档。Bob 对文档 m 签名之后所得的文档就是 $K_B^-(m)$。

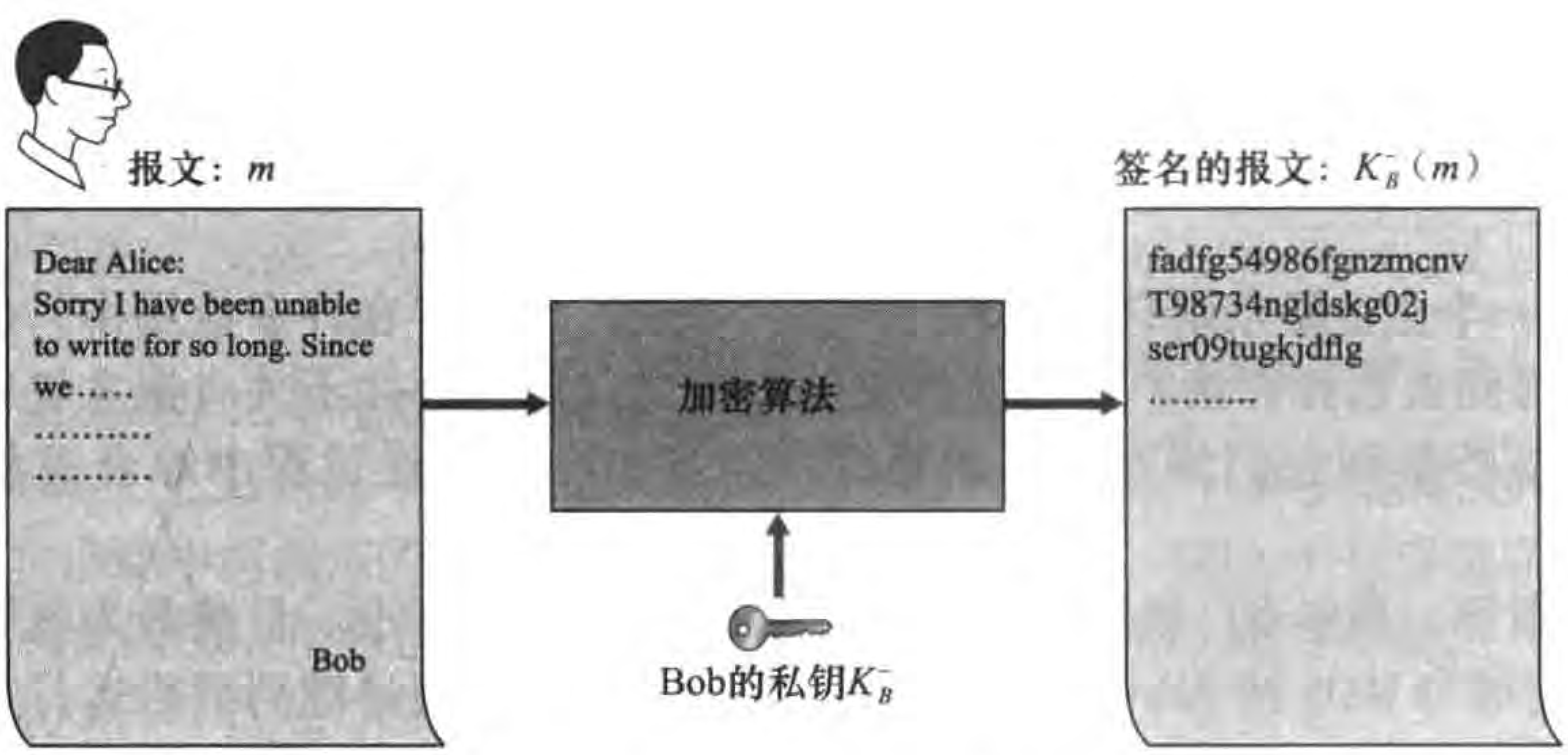


图 8-10 为一个文档生成一个数据签名

数字签名 $K_B^-(m)$ 是否满足了可鉴别、不可伪造的需求？假设 Alice 有 m 和$K_B^-(m)$。她要在法庭上证明（进行诉讼）Bob 确实签署过这个文档，他就是唯一能够签署该文档的人。Alice 持有 Bob 的公钥 K_B^+，并把它用于 Bob 的数字签名 $K_B^-(m)$，从而得到了文档 m。也就是说，Alice 计算 $K_B^+(K_B^-(m))$。瞧！在 Alice 经历了令人注目的慌乱后得到了 m，它与初始文档完全一致。然后，Alice 就可以论证仅有 Bob 能够签署这个文档，基于如下理由：

- 无论是谁签署这个报文，都必定在计算签名 $K_B^-(m)$ 过程中使用了 K_B^- 这个私钥，使 $K_B^+(K_B^-(m))=m$。
- 知道 K_B^- 这个私钥的唯一人只有 Bob。从 8.2 节我们对 RSA 的讨论中可知，知道公钥 K_B^+ 无助于得知私钥 K_B^- 的信息。因此，知道私钥 K_B^- 的人才是生成密钥对（K_B^+，K_B^-）的人，而这个人首当其冲就是 Bob。（注意到此处假设 Bob 没有把 K_B^- 泄露给任何人，也没有人从 Bob 处窃取到 K_B^-。）

注意到下列问题是重要的：如果源文档 m 被修改过，比如改成了另一个文档 m'，则 Bob 对 m 生成的签名对 m'无效，因为 $K_B^+(K_B^-(m))$ 不等于 m'。因此我们看到数字签名也

提供完整性，使得接收方验证该报文未被篡改，同时也验证了该报文的源。

对用加密进行数据签名的担心是，加密和解密的计算代价昂贵。给定加解密的开销，通过完全加密/解密对数据签名是杀鸡用牛刀。更有效的方法是将散列函数引入数字签名。8.3.2 节中讲过，一种散列算法取一个任意长的报文 m，计算生成该报文的一个固定长度的数据“指纹”，表示为 $H(m)$。使用散列函数，Bob 对报文的散列签名而不是对报文本身签名，即 Bob 计算 $K_B^-(H(m))$。因为 $H(m)$ 通常比报文 m 小得多，所以生成数字签名所需要的计算量大为降低。

在 Bob 向 Alice 发送一个报文的情况下，图 8-11 提供了生成数字签名的操作过程的概览。Bob 让他的初始长报文通过一个散列函数。然后他用自己的私钥对得到的散列进行数字签名。明文形式的初始报文连同已经数字签名的报文摘要（从此以后可称为数字签名）一道被发送给 Alice。图 8-12 提供了鉴别报文完整性的操作过程的概览。Alice 先把发送方的公钥应用于报文获得一个散列结果。然后她再把该散列函数应用于明文报文以得到第二个散列结果。如果这两个散列匹配，则 Alice 可以确信报文的完整性及其发送方。

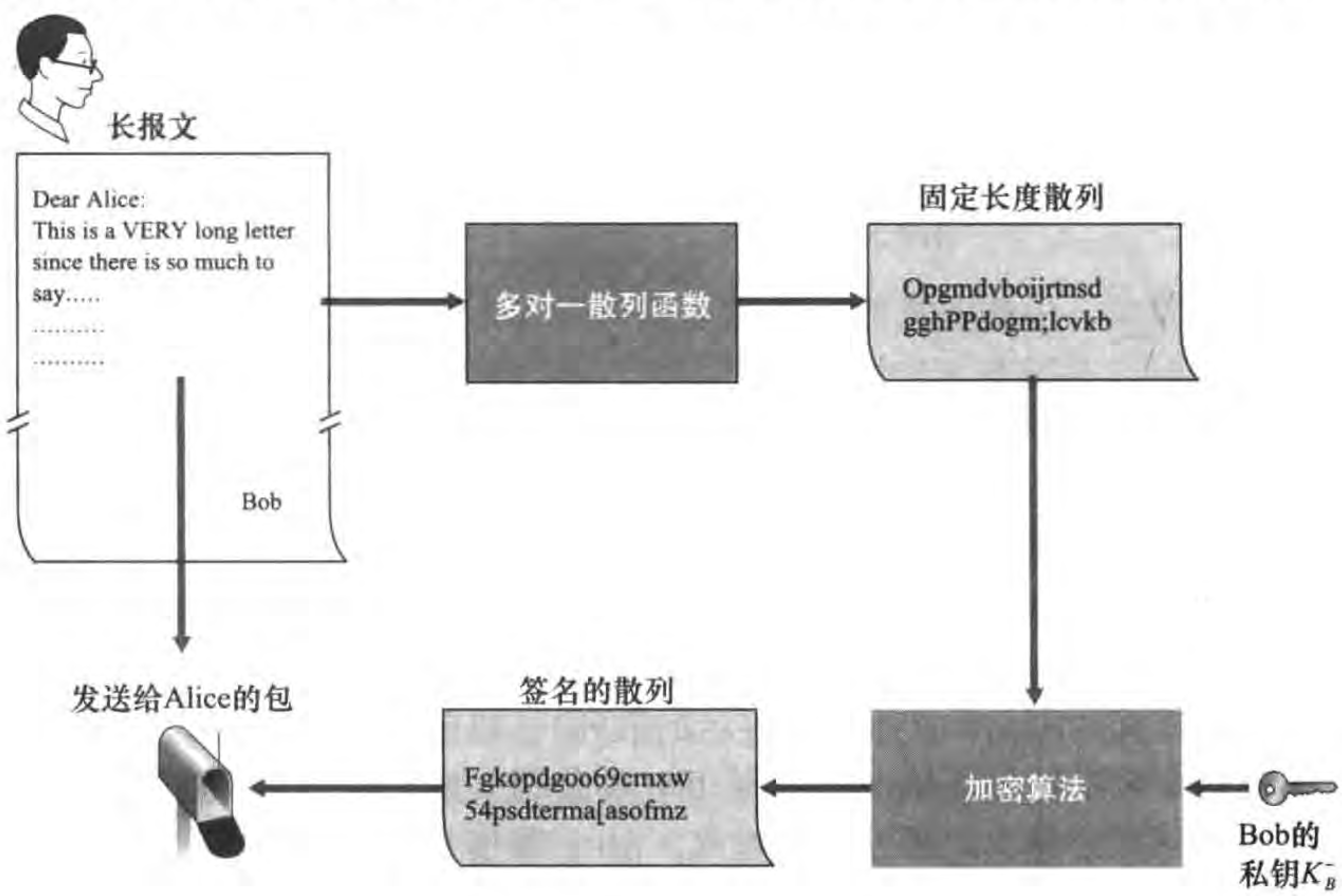


图 8-11 发送数字签名的报文

在继续学习之前，我们简要地将数字签名与 MAC 进行比较，尽管它们有类似之处，但也有重要的微妙差异。数字签名和 MAC 都以一个报文（或一个文档）开始。为了从该报文中生成一个 MAC，我们为该报文附加一个鉴别密钥，然后取得该结果的散列。注意到在生成 MAC 过程中既不涉及公开密钥加密，也不涉及对称密钥加密。为了生成一个数字签名，我们首先取得该报文的散列，然后用我们的私钥加密该报文（使用公钥密码）。因此，数字签名是一种“技术含量更高”的技术，因为它需要一个如后面描述的、具有认证中心支撑的公钥基础设施（PKI）。我们将在 8.4 节中看到，PGP 是一种流行的安全电子邮件系统，为了报文完整性而使用数字签名。我们已经看到了 OSPF 为了报文完整性而使用 MAC。我们将在 8.5 节和 8.6 节中看到 MAC 也能用于流行的运输层和网络层安全协议。

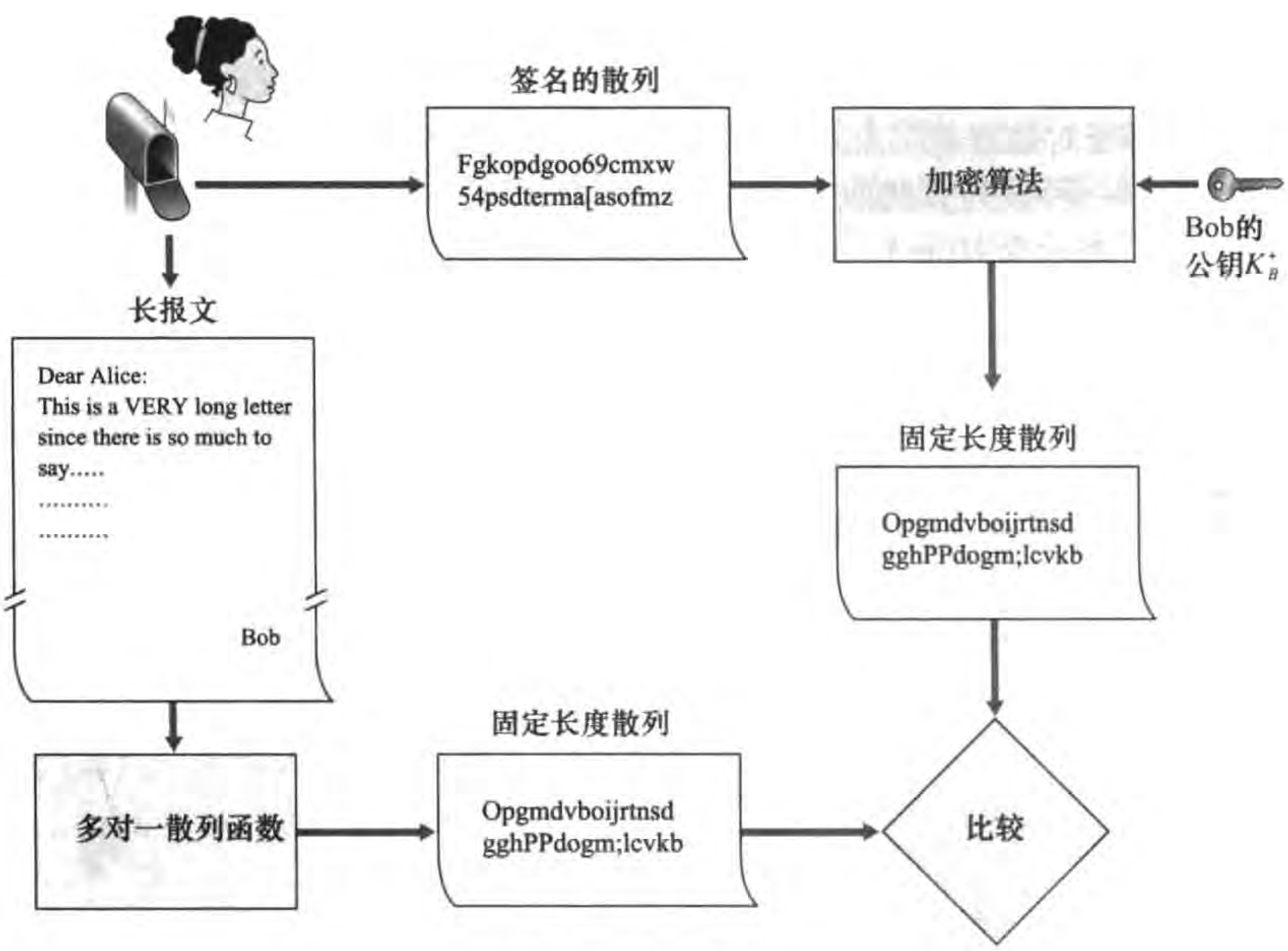


图 8-12 验证签名报文

公钥认证

数字签名的一个重要应用是**公钥认证**（public key certification），即证实一个公钥属于某个特定的实体。公钥认证用在许多流行的安全网络协议中，包括 IPsec 和 SSL。

为了深入理解这个问题，我们考虑一个因特网商务版本的经典的"比萨恶作剧"。假定 Alice 正在从事比萨派送业务，从因特网上接受订单。Bob 是一个爱吃比萨的人，他向 Alice 发送了一份包含其家庭地址和他希望的比萨类型的明文报文。Bob 在这个报文中也包含一个数字签名（即对原始明文报文的签名的散列），以向 Alice 证实他是该报文的真正来源。为了验证这个数字签名，Alice 获得了 Bob 的公钥（也许从公钥服务器或通过电子邮件报文）并核对该数字签名。通过这种方式，Alice 确信是 Bob 而不是某些青少年恶作剧者下的比萨订单。

在聪明的 Trudy 出现之前，这一切看起来进行得相当好。如图 8-13 中所示，Trudy 沉溺于一场恶作剧中。Trudy 向 Alice 发送一个报文，在这个报文中她说她是 Bob，给出了 Bob 家的地址并订购了一个比萨。在这个报文中，她也包括了她（Trudy）的公钥，虽然 Alice 自然地假定它就是 Bob 的公钥。Trudy 也附加了一个签名，但这是用她自己（Trudy）的私钥生成的。在收到该报文后，Alice 就会用 Trudy 的公钥（Alice 认为它是 Bob 的公钥）来解密该数字签名，并得到结论：这个明文报文确实是由 Bob 生成的。而当外送人员带着具有意大利辣香肠和凤尾鱼的比萨到达 Bob 家时，他会感到非常惊讶！

从这个例子我们看到，要使公钥密码有用，需要能够证实你具有的公钥实际上就是与你要进行通信的实体（人员、路由器、浏览器等）的公钥。例如，当 Alice 与 Bob 使用公钥密码通信时，她需要证实她假定是 Bob 的那个公钥确实就是 Bob 的公钥。

将公钥与特定实体绑定通常是由**认证中心**（Certification Authority，CA）完成的，CA 的职责就是使识别和发行证书合法化。CA 具有下列作用：

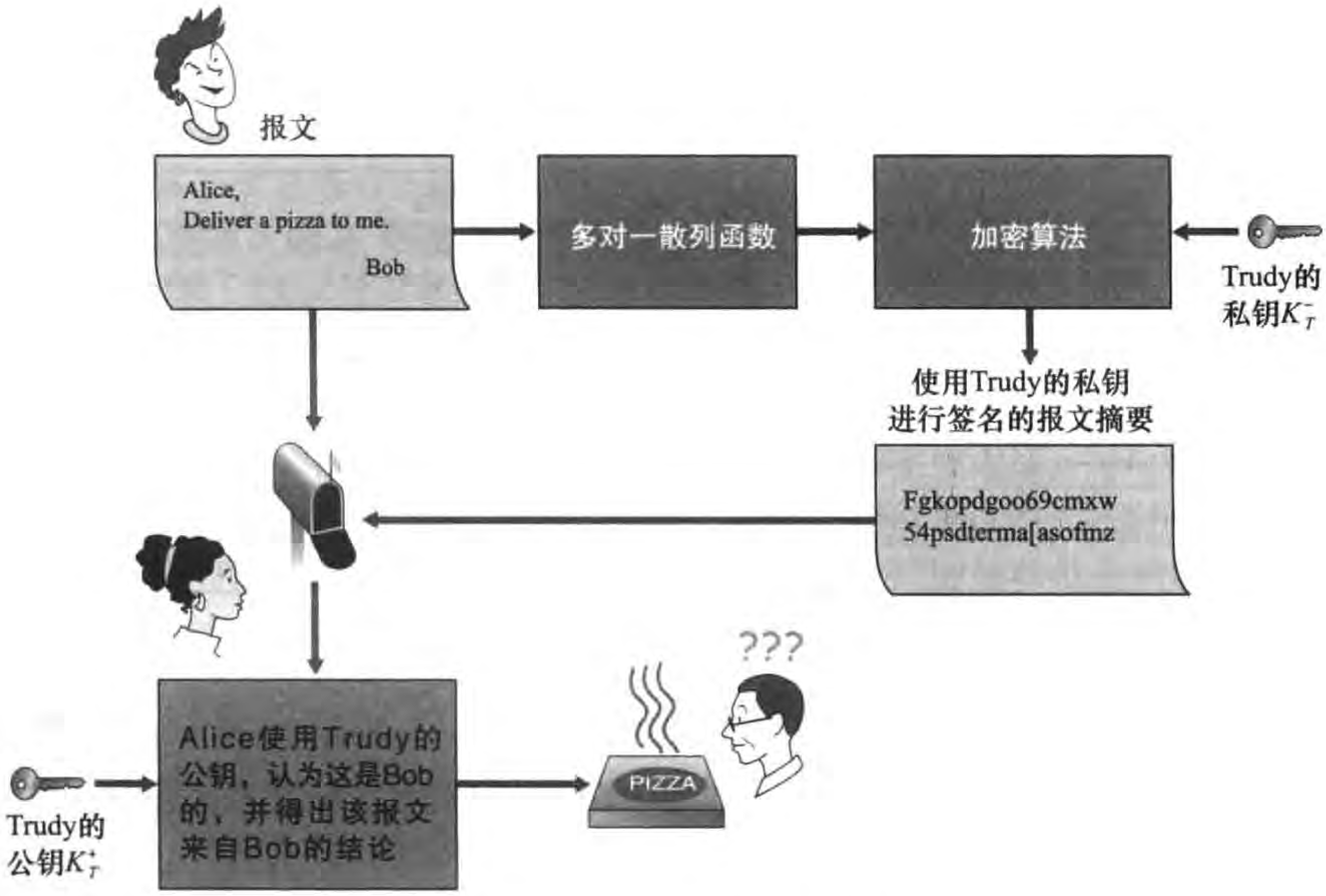


图 8-13　Trudy 用公钥密码冒充 Bob

1）CA 证实一个实体（一个人、一台路由器等）的真实身份。如何进行认证并没有强制的过程。当与一个 CA 打交道时，一方必须信任这个 CA 能够执行适当的严格身份验证。例如，如果 Trudy 走进名为 Fly-by-Night 的认证中心并只是宣称“我是Alice”，就可以得到该机构颁发的与 Alice 的身份相关联的证书的话，则人们不会对 Fly-by-Night 认证中心所签发的公钥证书有太多的信任。另一方面，人们可能愿意（或不愿意！）信任某个 CA，如果这个 CA 是联邦或州计划的一部分的话。你对与公钥相关联的身份的信任程度，仅能达到你对 CA 及其身份验证技术的信任程度。我们编织了多么混乱的信任关系网啊！

2）一旦 CA 验证了某个实体的身份，这个 CA 会生成一个将其身份和实体的公钥绑定起来的**证书**（certificate）。这个证书包含这个公钥和公钥所有者全局唯一的身份标识信息（例如，一个人的名字或一个 IP 地址）。由 CA 对这个证书进行数字签名。这些步骤显示在图 8-14 中。

我们现在来看怎样使用认证来对抗“比萨订购”中的恶作剧者（如 Trudy）和其他意外情况。Bob 下订单的同时，他也发送了其 CA 签署的证书。Alice 使用 CA 的公钥来核对 Bob 证书的合法性并提取 Bob 的公钥。

国际电信联盟（International Telecommunication Union，ITU）和 IETF 都研发了用于 CA 的系列标准。ITU X. 509 [ITU 2005a] 规定了证书的鉴别服务以及特定语法。[RFC 1422] 描述了安全

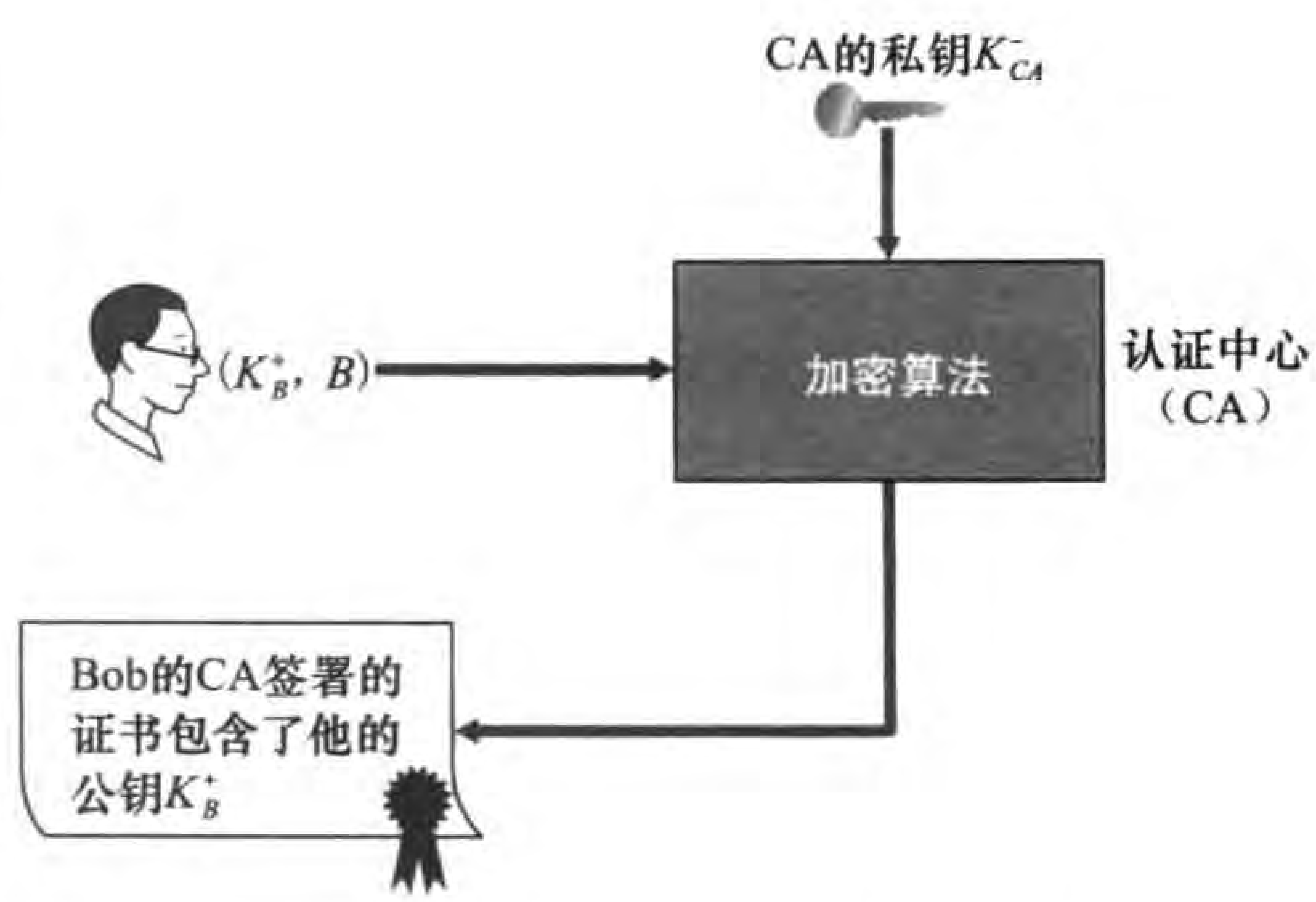


图 8-14　Bob 获得一份来自 CA 的证书

因特网电子邮件所用的基于 CA 的密钥管理。它和 X.509 兼容，但比 X.509 增加了密钥管理体系结构的创建过程和约定内容。表 8-4 显示了一份证书中的某些重要字段。

表 8-4 在 X.509 和 RFC 1422 公钥证书中的部分字段

字段名	描述
版本（Version）	X.509 规范的版本号
序列号（Serial number）	CA 发布的证书的独特标识符
签名（Signature）	规定了由 CA 所用的对该证书签名的算法
颁发者名称（Issuer name）	发行该证书的 CA 的标识符，用的是区别名（DN）格式［RFC 4514］
有效期（Validity period）	证书合法性开始和结束的时间范围
主题名（Subject name）	其公钥与该证书相联系的实体标识符，用 DN 格式
主题公钥（Subject public key）	该主题的公钥以及该公钥使用的公钥算法（及其参数）的指示

8.4 端点鉴别

端点鉴别（end-point authentication）就是一个实体经过计算机网络向另一个实体证明其身份的过程，例如一个人向某个电子邮件服务器证明其身份。作为人类，我们通过多种方式互相鉴别：见面时我们互相识别对方的面容，打电话时我们分辨对方的声音，海关的检查官员通过护照上的照片对我们进行鉴别。

在本节中，我们讨论经网络通信的双方如何能够鉴别彼此。此处我们重点关注当通信实际发生时鉴别“活动的”实体。一个具体的例子是一个用户向某电子邮件服务器鉴别他或她自己。这与证明在过去的某点接收到的报文确实来自声称的发送方稍有不同，如 8.3 节所述。

当经网络进行鉴别时，通信各方不能依靠生物信息比如外表、声波纹等进行身份鉴别。的确，我们会在后面的实例研究中看到，诸如路由器、客户/服务器进程等网络元素通常必须相互鉴别。此处，鉴别应当在报文和数据交换的基础上，作为某**鉴别协议**（authentication protocol）的一部分独立完成。鉴别协议通常在两个通信实体运行其他协议（例如，可靠数据传输协议、路由选择信息交换协议或电子邮件协议）之前运行。鉴别协议首先建立相互满意的各方的标识；仅当鉴别完成之后，各方才继续下面的工作。

同第 3 章中我们阐释可靠数据传输协议（rdt）的情况类似，我们发现阐释各种版本的鉴别协议——我们将称为 ap（authentication protocol）——是有启发的，并随着我们学习的深入指出各个版本的漏洞。（如果你喜欢这种逐步式的设计演化，你也许喜欢看［Bryant 1988］，这本书虚构了开放网络鉴别系统的设计者间的故事，以及他们对许多相关奇妙问题的发现。）

我们假设 Alice 要向 Bob 鉴别她自己的身份。

8.4.1 鉴别协议 ap1.0

也许我们能够想象出的最简单的鉴别协议就是：Alice 直接发送一个报文给 Bob，说她就是 Alice。这个协议如图 8-15所示。这个协议的缺陷是明显的，即 Bob 无法判断发送报文“我是

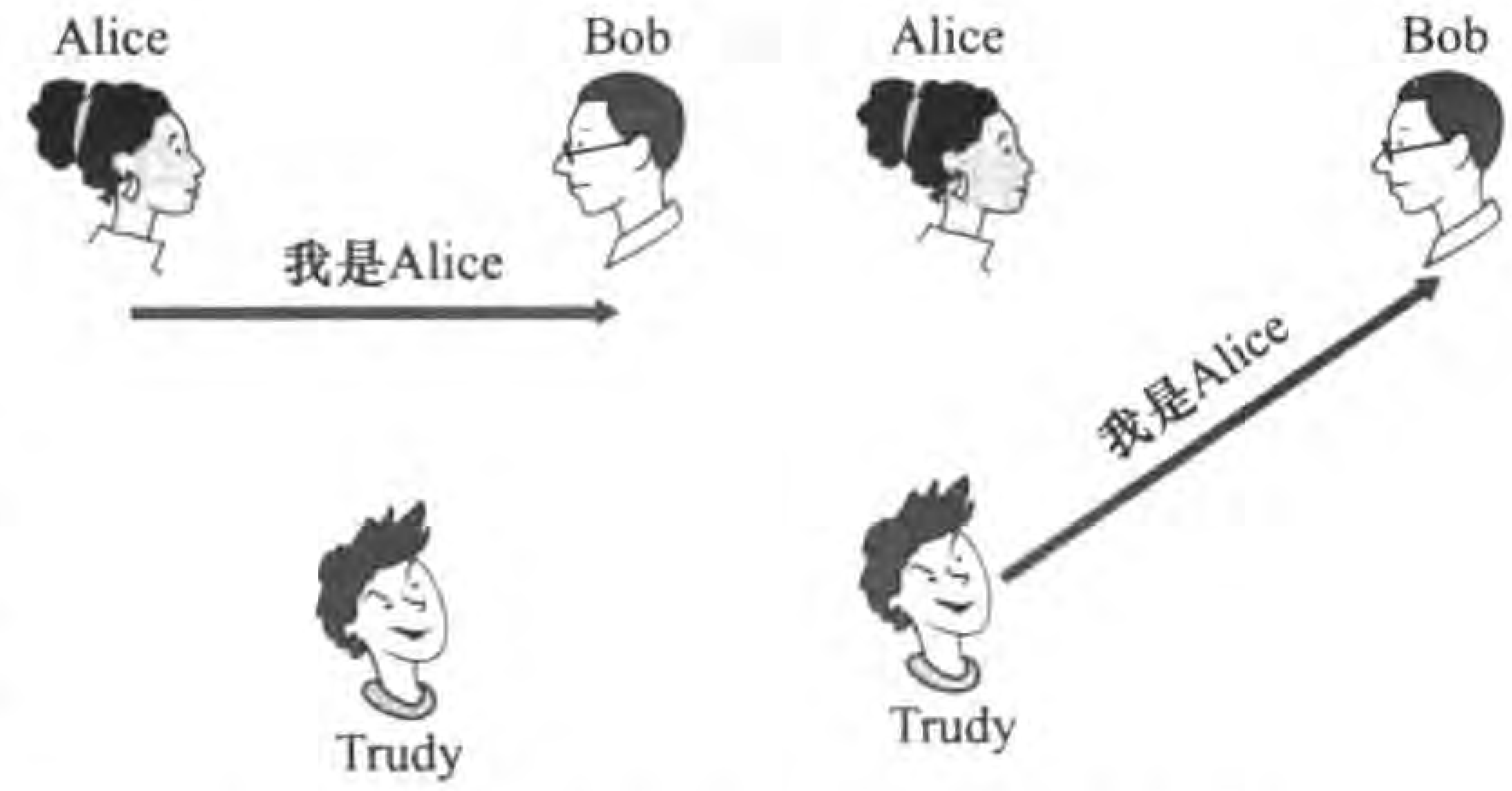


图 8-15 协议 ap1.0 和一种失败的情况

Alice”的人确实就是 Alice。例如，Trudy（入侵者）也可以发送这样的报文。

8.4.2 鉴别协议 ap2.0

如果 Alice 有一个总是用于通信的周知网络地址（如一个 IP 地址），则 Bob 能够试图通过验证携带鉴别报文的 IP 数据报的源地址是否与 Alice 的周知 IP 地址相匹配来进行鉴别。在这种情况下，Alice 就可被鉴别了。这可能阻止对网络一无所知的人假冒 Alice，但是它却不能阻止决定学习本书的学生或许多其他人！

根据我们学习的网络层和数据链路层的知识，我们就会知道做下列事情并不困难（例如，如果一个人能够访问操作系统代码并能构建自己的操作系统内核——比如 Linux 和许多其他免费可用的操作系统）：生成一个 IP 数据报，并在 IP 数据报中填入我们希望的任意源地址（比如 Alice 的周知 IP 地址），再通过链路层协议把生成的数据报发送到第一跳路由器。此后，具有不正确源地址的数据报就会忠实地向 Bob 转发。这种方法显示在图 8-16中，它是 IP 哄骗的一种形式。如果 Trudy 的第一跳路由器被设置为只转发包含 Trudy 的 IP 源地址的数据报，就可以避免 IP 哄骗［RFC 2827］。然而，这一措施并未得到广泛采用或强制实施。Bob 可能因为假定 Trudy 的网络管理员（这个管理员可能就是 Trudy 自己）已经配置 Trudy 的第一跳路由器，使之只能转发适当地址的数据报而被欺骗。

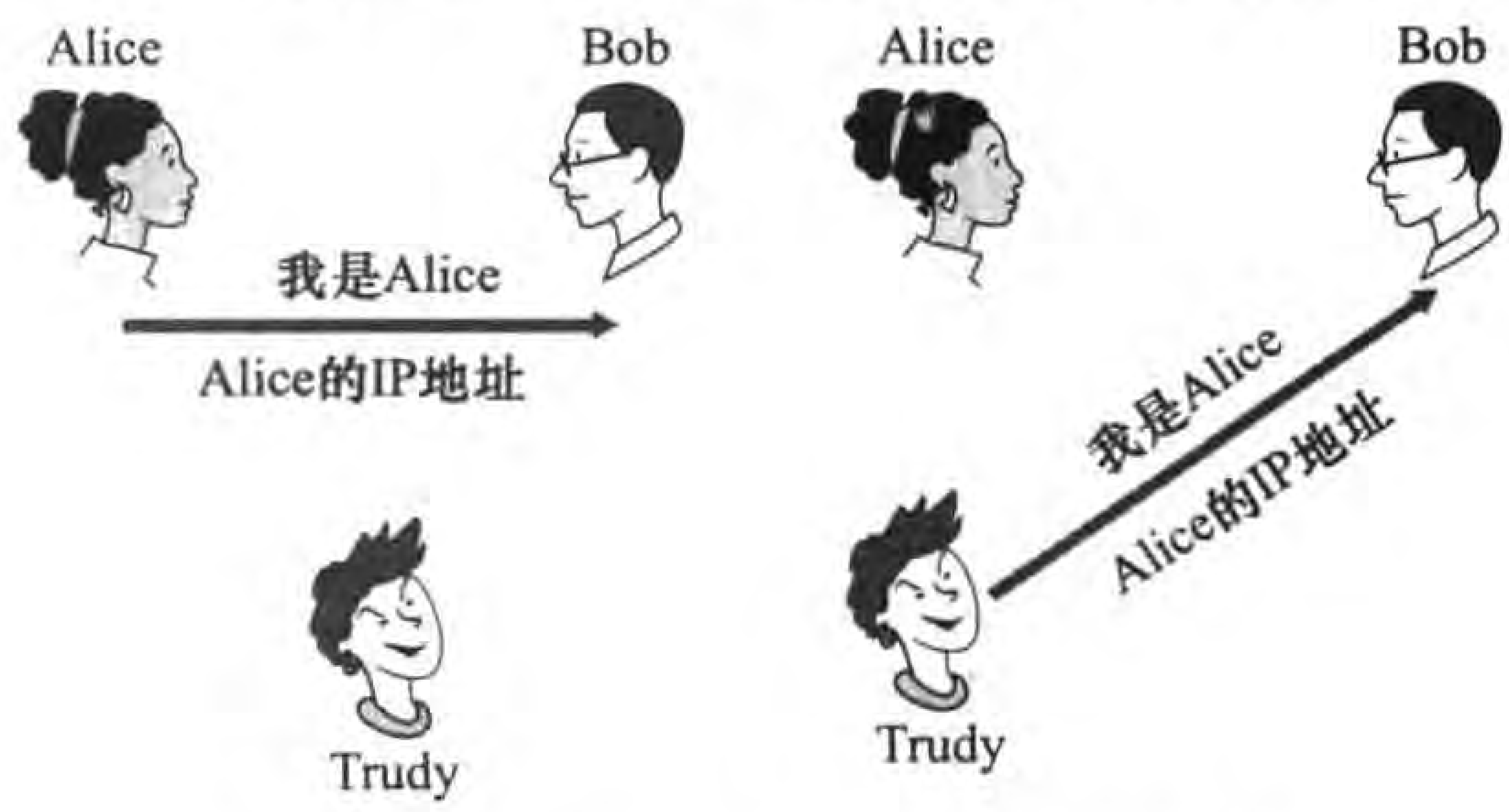


图 8-16 协议 ap2.0 和一种失败的情况

8.4.3 鉴别协议 ap3.0

进行鉴别的一种经典方法是使用秘密口令。口令是鉴别者和被鉴别者之间的一个共享秘密。Gmail、Telnet、FTP 和许多其他服务使用口令鉴别。在协议 ap3.0 中，Alice 因此向 Bob 发送其秘密口令，如图 8-17 所示。

由于口令的广泛使用，我们也许猜想协议 ap3.0 相当安全。如果这样想，我们就错了！这里的安全性缺陷相当明显：如果 Trudy 窃听了 Alice 的通信，则可得到 Alice 的口令。为了使你认识到这种可能性，考虑这样的事实，当你 Telnet 到另一个机器上并登录时，登录口令未加密就发送到了 Telnet 服务器。连接到 Telnet 客户或服务器 LAN 的某个人都可能嗅探（sniff）（读并存储）在局域网上传输的所有数据分组，并因此窃取到该注册口令。实际上，这是一种窃取口令的周知方法（例如，参见［Jimenez 1997］）。这样的威胁显然是真实存在的，所以协议 ap3.0 明显也不可行。

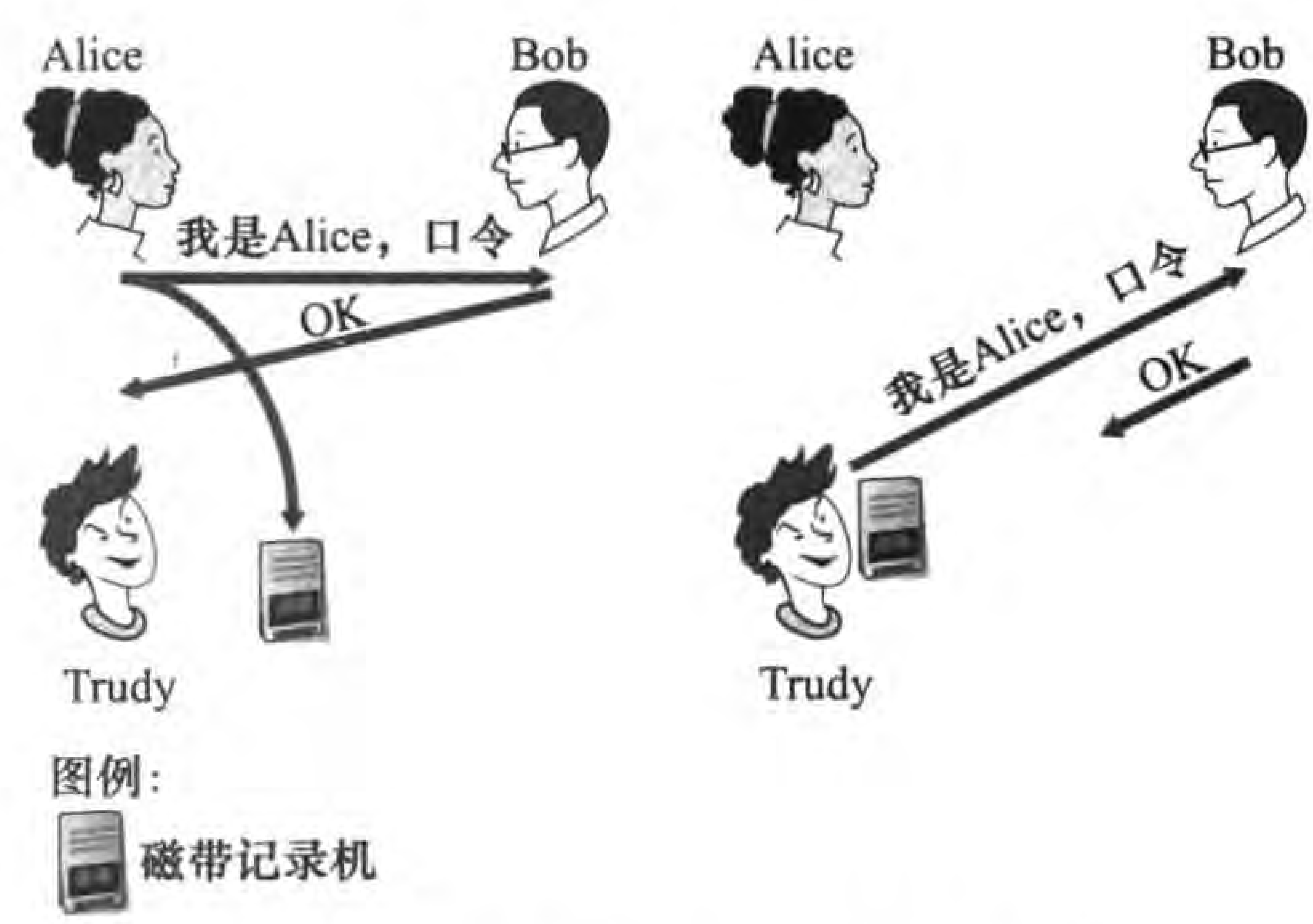


图 8-17 协议 ap3.0 和一种失败的情况

8.4.4 鉴别协议 ap3.1

我们完善协议 ap3.0 的下一个想法自然就是加密口令了。通过加密口令，我们能够防止 Trudy 得知 Alice 的口令。如果我们假定 Alice 和 Bob 共享一个对称秘密密钥 K_{A-B}，则 Alice 可以加密口令，并向 Bob 发送其识别报文“我是 Alice”和加密的口令。Bob 则解密口令，如果口令正确则鉴别了 Alice。因为 Alice 不仅知道口令，而且知道用于加密口令的共享秘密密钥值，Bob 才可以轻松地鉴别 Alice 的身份。我们称这个协议为 ap3.1。

尽管协议 ap3.1 确实防止了 Trudy 得知 Alice 的口令，此处使用密码术并不能解决鉴别问题。Bob 受制于**回放攻击**（playback attack）：Trudy 只需窃听 Alice 的通信，并记录下该口令的加密版本，并向 Bob 回放该口令的加密版本，以假装她就是 Alice。协议 ap3.1 中加密口令的使用，并未使它比图 8-17 中的协议 ap3.0 的局面有明显改观。

8.4.5 鉴别协议 ap4.0

图 8-17 中的失败的情况是因为 Bob 不能区分 Alice 的初始鉴别报文和后来入侵者回放的 Alice 的初始鉴别报文所致。也就是说，Bob 无法判断 Alice 是否还活跃（即当前是否还在连接的另一端），或他接收到的报文是否就是前面鉴别 Alice 时录制的回放。观察力极强的读者会记起 TCP 的三次握手协议需要处理相同的问题，如果接收的 SYN 报文段来自较早连接的一个 SYN 报文段的旧副本（重新传输）的话，TCP 连接的服务器一侧不会接受该连接。TCP 服务器一侧如何解决“判断客户是否真正还活跃”的问题呢？它选择一个很长时间内都不会再次使用的初始序号，然后把这个序号发给客户，然后等待客户以包含这个序号的 ACK 报文段来响应。此处我们能够为鉴别目的采用同样的思路。

不重数（nonce）是在一个协议的生存期中只使用一次的数。也就是说，一旦某协议使用了一个不重数，就永远不会再使用那个数字了。协议 ap4.0 以如下方式使用一个不重数：

1）Alice 向 Bob 发送报文“我是 Alice”。

2）Bob 选择一个不重数 R，然后把这个值发送给 Alice。

3）Alice 使用她与 Bob 共享的对称秘密密钥 K_{A-B}来加密这个不重数，然后把加密的不重数 $K_{A-B}(R)$ 发回给 Bob。与在协议 ap3.1 中一样，由于 Alice 知道 K_{A-B}并用它加密一个值，就使得 Bob 知道收到的报文是由 Alice 产生的。这个不重数用于确定 Alice 是活跃的。

4）Bob 解密接收到的报文。如果解密得到的不重数等于他发送给 Alice 的那个不重数，则可鉴别 Alice 的身份。

协议 ap4.0 如图 8-18 所示。通过使用这个在生存期中只出现一次的值 R，然后核对返回的值 $K_{A-B}(R)$，Bob 能够确定两点：Alice 是她所声称的那个人（因为她知道加密 R 所需的秘密密钥），Alice 是活跃的（因为她已经加密了 Bob 刚刚产生的不重数 R）。

不重数和对称密钥密码体制的使用形成了 ap4.0 的基础。一个自然的问题是，我们是否能够使用不重数和公开密钥密码体制（而不是对称密钥密码体制）来解决鉴别问题？这个问题将在本章后面的习题中进行探讨。

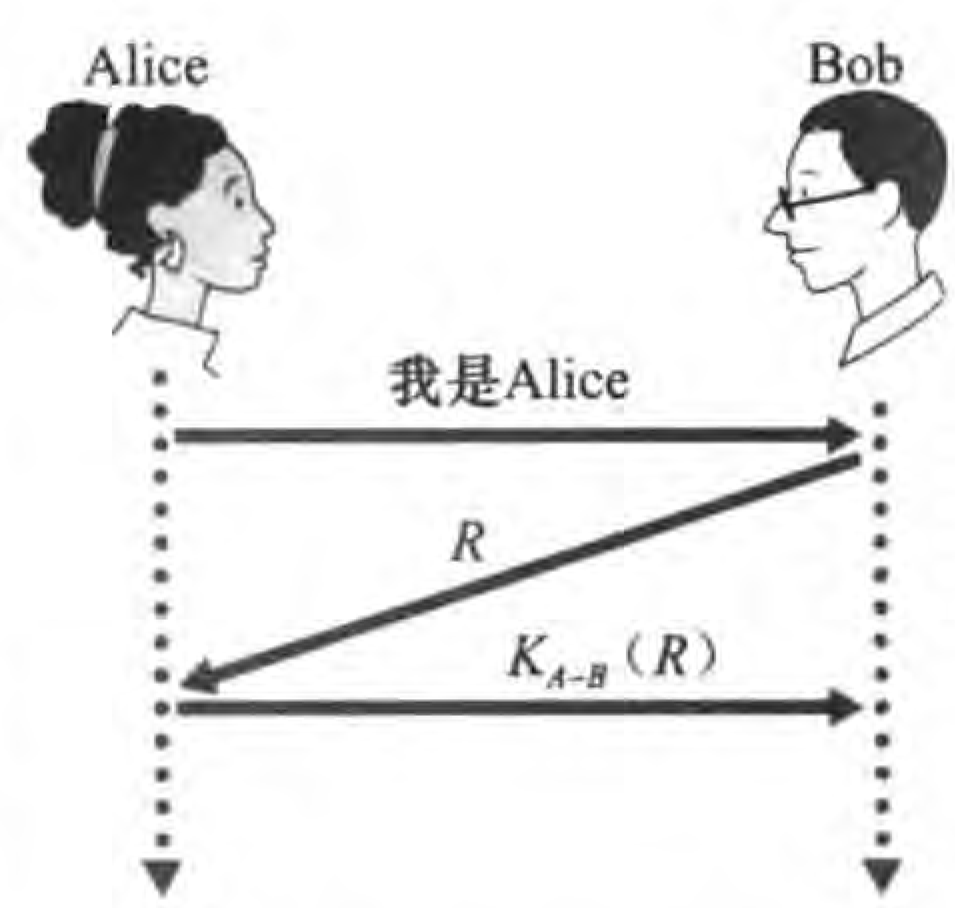


图 8-18 协议 ap4.0

8.5 安全电子邮件

在前面的各节中，我们分析了网络安全中的基本问题，包括对称密钥密码体制和公开密钥密码体制、端点鉴别、密钥分发、报文完整性和数字签名。我们现在着手研究如何使用这些工具在因特网中提供安全性。

有趣的是，为因特网协议栈上面4层的任一层提供安全性服务是可能的。当为某一特定的应用层协议提供安全性时，则使用这一协议的应用程序将能得到一种或多种安全服务，诸如机密性、鉴别或完整性。为某一运输层协议提供安全性时，则所有使用这一协议的应用程序都可以得到该运输层协议所提供安全性服务。在基于主机到主机的网络层提供安全性时，则所有运输层报文段（当然也包括所有应用层数据）都可以得到该网络层所提供的安全服务。当基于一条链路提供安全性时，则经过这个链路传输的所有帧中的数据都得到了该链路提供的安全性服务。

在8.5~8.8节中，我们考察了如何在应用层、运输层、网络层和数据链路层中使用这些安全性工具。为了与本书的整体框架保持一致，我们从协议栈的顶层开始，讨论在应用层的安全性。我们的方法是使用特定的应用程序如电子邮件，作为应用层安全性的一个学习案例。然后我们沿协议栈向下，分析SSL协议（它在运输层提供安全性）、IPsec协议（它在网络层提供安全性），以及IEEE 802.11无线局域网协议的安全性。

你可能会感到奇怪：为什么要在因特网的多个层次上提供安全性功能呢？仅在网络层提供安全性功能并加以实施还不足够吗？对这个问题有两个答案。首先，尽管可以通过加密数据报中的所有数据（即所有的运输层报文段），以及通过鉴别所有数据报的源IP地址，在网络层能够提供“地毯式覆盖”安全性，但是却并不能提供用户级的安全性。例如，一个商业站点不能依赖IP层安全性来鉴别一个在该站点购买商品的顾客。因此，此处除了较低层的地毯式覆盖安全性外，还需要更高层的安全性功能。第二，在协议栈的较高层上部署新的因特网服务（包括安全性服务）通常较为容易。而等待在网络层上广泛地部署安全性，可能还需要未来若干年才能解决，许多应用程序的开发者“着手做起来”，并在他们中意的应用程序中引入安全性功能。一个典型的例子就是PGP(Pretty Good Privacy)，它提供了安全电子邮件（将在本节后面讨论）。由于只需要客户和服务器应用程序代码，PGP是第一个在因特网上得到广泛应用的安全性技术。

8.5.1 安全电子邮件

我们现在使用8.2节和8.3节的密码学原则来生成一个安全电子邮件系统。我们以递进的方式来产生这个高层设计，每一步引入一些新安全性服务。当设计安全电子邮件系统时，我们需要记住最初在8.1节中所介绍的那个有趣的例子，即Alice和Bob之间的风流韵事。设想一下Alice发送一个电子邮件报文给Bob，而Trudy试图入侵的情况。

在做出为Alice和Bob设计一个安全电子邮件系统的努力之前，我们应当首先考虑他们最为希望的安全特性是什么。重中之重是*机密性*。正如8.1节讨论的那样，Alice或Bob都不希望Trudy阅读到Alice所发送的电子邮件报文。Alice和Bob最希望在该电子邮件系统中看到的第二种特性是具备*发送方鉴别*。特别是，当Bob收到这样的报文“I don't love you anymore. I never want to see you again. Formerly yours, Alice（我不再爱你了。我再也不想看到你了。Alice）”时，Bob自然而然地要确定这个报文确实来自Alice，而非Trudy发

送的。另外，这两个情人欣赏的另一种特性是报文完整性，也就是说，确保 Alice 所发的报文在发送给 Bob 的过程中没有被改变。最后，电子邮件系统应当提供接收方鉴别；即 Alice 希望确定她的确正在向 Bob 发信，而不是向假冒 Bob 的其他人（如 Trudy）发信。

因此我们从处理最为关注的机密性开始。提供机密性的最直接方式是 Alice 使用对称密钥技术（如 DES 或 AES）加密所要传输的报文，而 Bob 则在接收时对报文解密。如 8.2 节讨论的那样，如果对称密钥足够长，且仅有 Alice 和 Bob 拥有该密钥，则其他人（包括 Trudy）要想读懂这条报文极为困难。尽管这种方法直截了当，但因为仅有 Alice 和 Bob 具有该密钥的副本，这使得分发对称密钥非常困难（我们在 8.2 节中讨论过）。因此我们自然就考虑用其他方法——公开密钥密码（例如使用 RSA）。在公开密钥方法中，Bob 使得他的公钥为公众所用（例如，从一台公钥服务器或其个人网页上得到），Alice 用 Bob 的公钥加密她的报文，然后向 Bob 的电子邮件地址发送该加密报文。当 Bob 接收到这个报文时，只需用他的私钥即可解密之。假定 Alice 确定得到的公钥是 Bob 的公钥，这种方法是提供所希望的机密性的极好方法。然而，存在的一个问题是公开密钥加密的效率相对低下，尤其对于长报文更是如此。

为了克服效率问题，我们利用了会话密钥（在 8.2.2 节中讨论过）。具体来说：①Alice选择一个随机对称会话密钥 K_S；②用这个对称密钥加密她的报文 m；③用 Bob 的公钥 K_B^+ 加密这个对称密钥；④级联该加密的报文和加密的对称密钥以形成一个“包”；⑤向 Bob 的电子邮件地址发送这个包。这些过程显示在图 8-19 中（在这张图和下一张图中，带圈的“+”表示级联，带圈的“-”表示级联的分解）。当 Bob 接收到这个包时：①他使用其私钥 K_B^- 得到对称密钥 K_S；②使用这个对称密钥 K_S 解密报文 m。

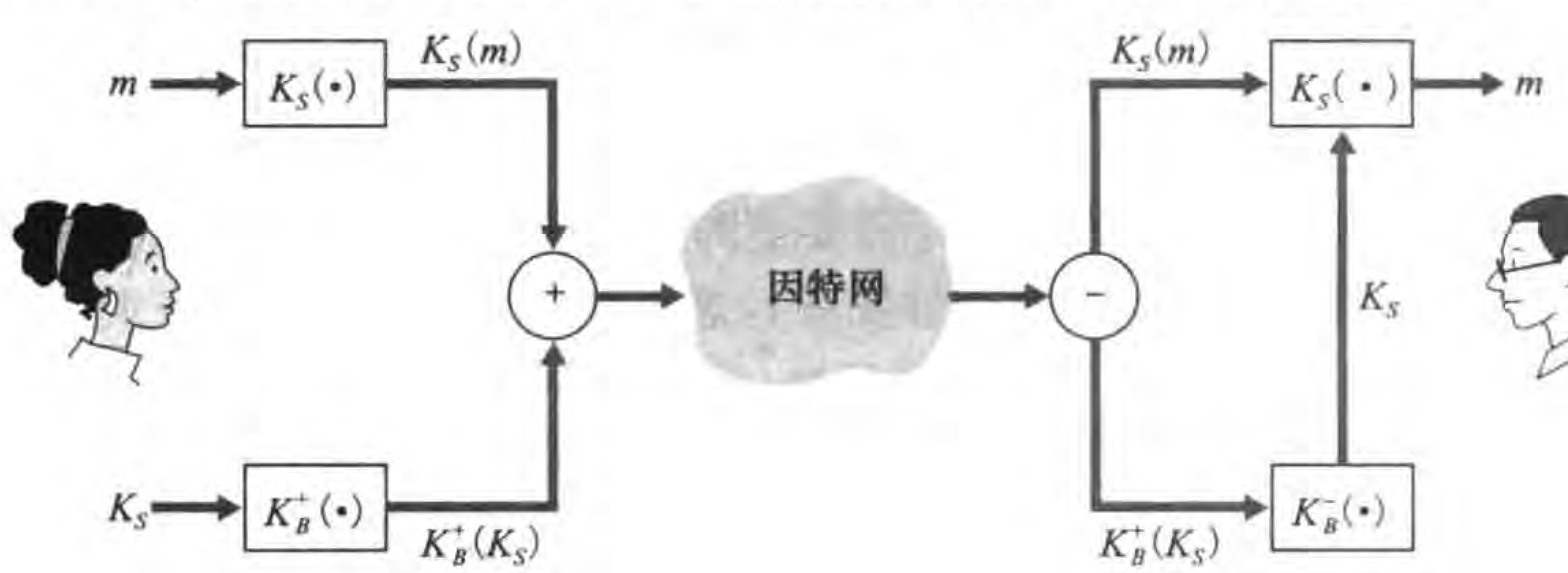


图 8-19　Alice 使用一个对称会话密钥 K_S 向 Bob 发送一个安全电子邮件

设计完提供机密性的安全电子邮件系统后，现在我们设计另一个可以提供发送方鉴别和报文完整性的系统。我们暂且假设 Alice 和 Bob 目前不关心机密性（他们要和其他人分享他们的爱情!），只关心发送方鉴别和报文完整性。为了完成这个任务，我们使用如 8.3 节所描述的数字签名和报文摘要。具体说来：①Alice 对她要发送的报文 m 应用一个散列函数 H（例如 MD5），从而得到一个报文摘要；②用她的私钥 K_A^- 对散列函数的结果进行签名，从而得到一个数字签名；③把初始报文（未加密）和该数字签名级联起来生成一个包；④向 Bob 的电子邮件地址发送这个包。当 Bob 接收到这个包时：①他将 Alice 的公钥 K_A^+ 应用到被签名的报文摘要上；②将该操作的结果与他自己对该报的散列 H 进行比较。在图 8-20 中阐述了这些步骤。如 8.3 节中所讨论，如果这两个结果相同，则 Bob 完全可以确信这个报文来自 Alice 且未被篡改。

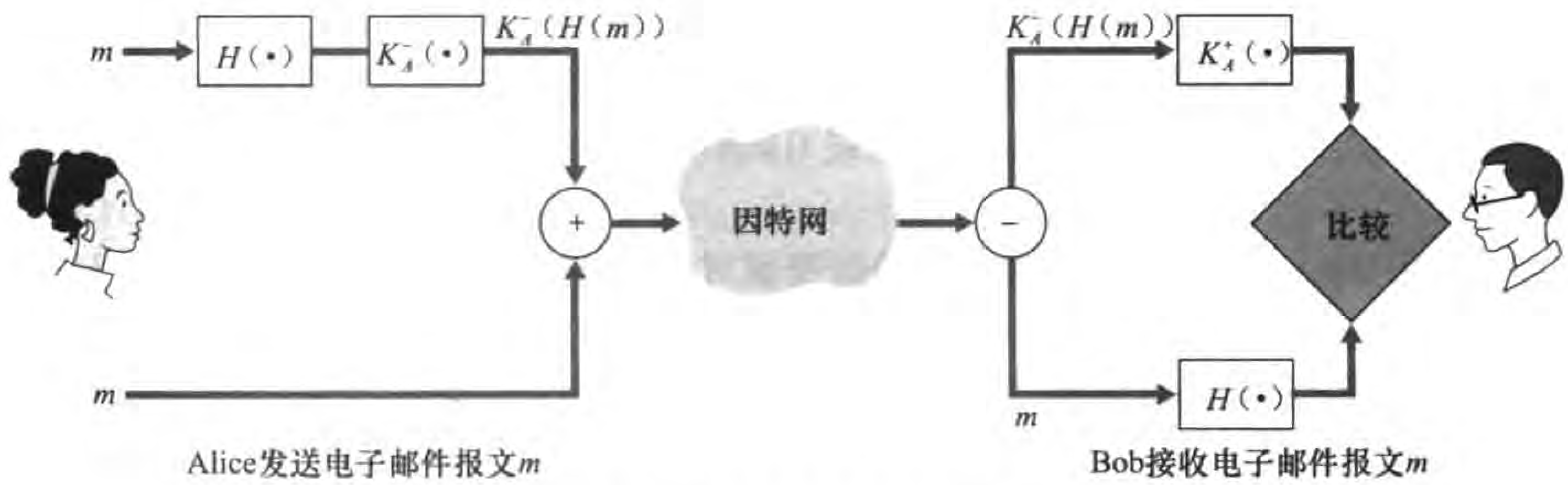


图 8-20 使用散列函数和数字签名来提供发送方鉴别和报文完整性

现在我们考虑设计一个提供机密性、发送方鉴别和报文完整性的电子邮件系统。这可以通过把图 8-19 和图 8-20 中的过程结合起来而实现。Alice 首先生成一个预备包，它与图 8-20 中的包完全相同，其中包含她的初始报文和该报文数字签名过的散列。然后 Alice 把这个预备包看作一个报文，再用图 8-19 中的发送方的步骤发送这个新报文，即生成一个新包发给 Bob。Alice 所做的这些步骤如图 8-21 所示。当 Bob 接收到这个包后，他首先应用图 8-19 中他这一侧的步骤，然后再应用图 8-20 中他这一侧的步骤。应当明确这一设计的目标是提供机密性、发送方鉴别和报文完整性。注意到在这一方案中，Alice 两次使用了公开密钥密码：一次用了她的私钥，另一次用了 Bob 的公钥。类似地，Bob 也两次使用了公开密钥密码：一次用了他的私钥，一次用了 Alice 的公钥。

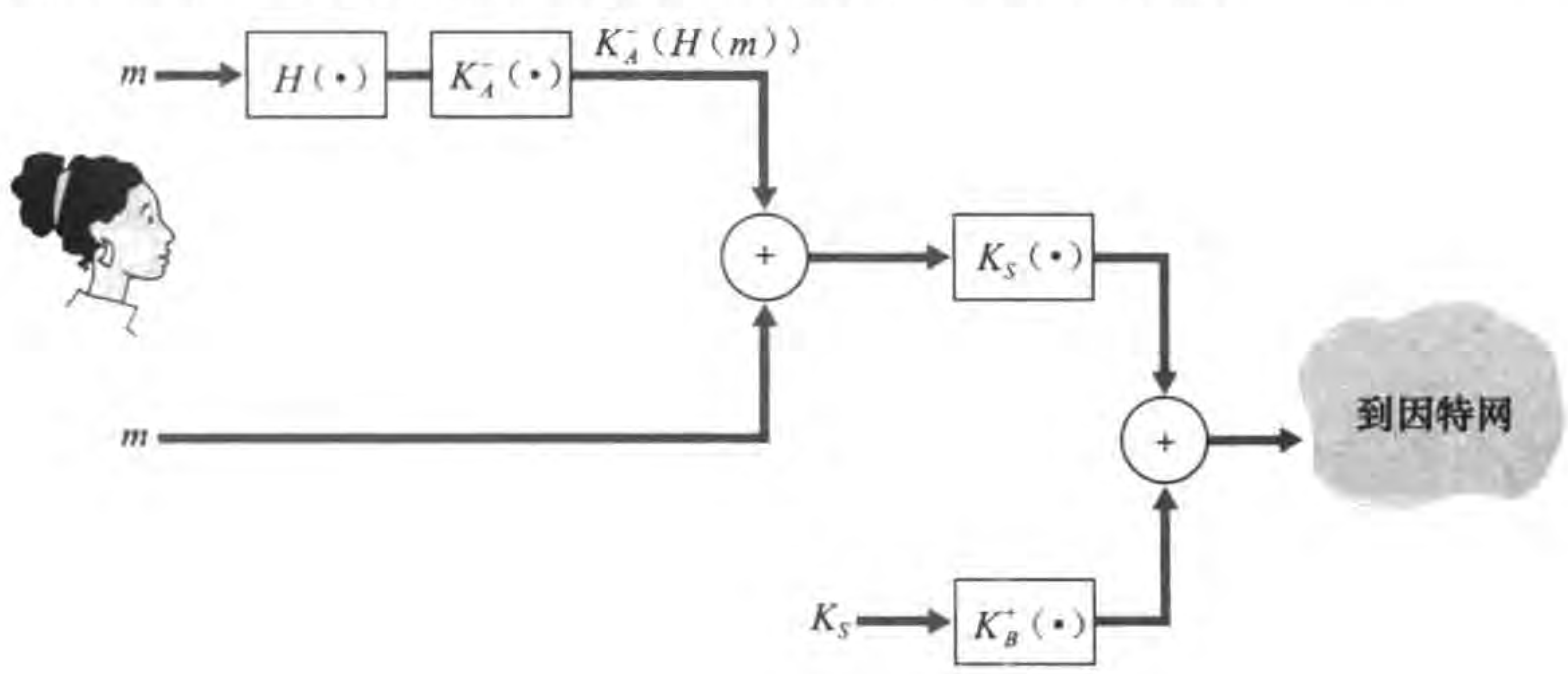


图 8-21 Alice 使用对称密钥密码、公开密钥密码、散列函数和数字签名来提供安全性、发送方鉴别和报文完整性

图 8-21 所示的安全电子邮件系统可能在大多数情况下都能为大多数电子邮件用户提供满意的安全性。但是仍有一个重要的问题没有解决。图 8-21 中的设计要求 Alice 获得 Bob 的公钥，也要求 Bob 获得 Alice 的公钥。但这些公钥的分发并不是一个小问题。例如，Trudy 可能假冒 Bob，发给 Alice 她自己的公钥，并告诉 Alice 这个公钥是 Bob 的公钥，使得 Trudy 就能接收到 Alice 发给 Bob 的报文。如我们在 8.3 节所学，安全地分发公钥的一种常用方法是通过 CA 验证该公钥。

8.5.2 PGP

Philip Zimmermann 于 1991 年所写的 PGP（Pretty Good Privacy）是电子邮件加密方案的一个范例［PGPI 2016］。在公共领域中有各种版本的 PGP 可供使用；例如，你能够在国

际 PGP 的主页上为你喜爱的平台找到 PGP 软件以及许多有趣的读物［PGPI 2016］。PGP 的设计在本质上和图 8-21 中所示的设计相同。PGP 软件的不同版本使用 MD5 或使用 SHA 来计算报文摘要；使用 CAST、三重 DES 或 IDEA 进行对称密钥加密；使用 RSA 进行公开密钥加密。

安装 PGP 时，软件为用户产生一个公开密钥对。该公钥能被张贴到用户的网站上或放置在某台公钥服务器上。私钥则使用用户口令进行保护。用户每次访问私钥时都要输入这个口令。PGP 允许用户选择是否对报文进行数字签名、加密报文，或同时进行数字签名和加密。图 8-22 显示了一个 PGP 签名的报文。这个报文在 MIME 首部之后出现。报文中的加密数据为 $K_A^-(H(m))$，即数字签名的报文摘要。如我们上述讨论，Bob 为了验证报文的完整性，需要得到 Alice 的公钥。

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash:  SHA1
Bob:
Can I see you tonight?
Passionately yours, Alice
-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset:  noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
-----END PGP SIGNATURE-----
```

图 8-22　PGP 签名报文

图 8-23 显示了一个秘密 PGP 报文。这个报文也出现在 MIME 首部之后。当然，明文报文不包括在这个秘密电子邮件报文中。当一个发送方（例如 Alice）要确保机密性和完整性时，PGP 在如图 8-23 所示的报文中包含一个类似于图 8-22 中的报文。

```
-----BEGIN PGP MESSAGE-----
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=lKhnMikLo0+l/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmKlok8iy6gThlp
-----END PGP MESSAGE-----
```

图 8-23　一个秘密 PGP 报文

PGP 也提供了一种公钥认证机制，但是这种机制与更为传统的 CA 差异很大。PGP 公钥由一个可信 Web 验证。当 Alice 相信一个密钥/用户名对确实匹配时，她自己就可以验证这一密钥/用户名对。此外，PGP 允许 Alice 为她所信任的用户鉴别更多密钥提供担保。一些 PGP 用户通过保持密钥签署方（key-signing party）互相签署对方的密钥。用户实际走到一起，交换公钥，并用自己的私钥对对方的公钥签名来互相验证密钥。

8.6　使 TCP 连接安全：SSL

在前一节中，我们看到对一个特定的应用（即电子邮件），密码技术是怎样提供机密性、数据完整性和端点鉴别的。在这一节中，我们在协议栈中向下一层，考察密码技术如何用安全性服务加强 TCP，该安全性服务包括机密性、数据完整性和端点鉴别。TCP 的这种强化版本通常被称为**安全套接字层**（Secure Socket Layer，SSL）。SSL 版本 3 的一个稍加修改的版本被称为**运输层安全性**（Transport Layer Security，TLS），已经由 IETF 标准化［RFC 4346］。

SSL 最初由 Netscape 设计，而使 TCP 安全隐含的基本思想先于 Netscape 的工作（例如，参见［Woo 1994］）。由于 SSL 的崭露头角，它已经得到了广泛部署。SSL 得到了所有流行 Web 浏览器和 Web 服务器的支持，并基本上被用于所有因特网商业站点（包括 Ama-

zon、eBay、Yahoo!、MSN 等等)。每年经 SSL 花费了数百亿美元。事实上，如果你使用信用卡通过因特网购买任何东西的话，在你的浏览器和服务器之间的通信几乎一定使用了 SSL。(当你使用浏览器时，若 URL 以 https：开始而不是以 http 开始，就能认定正在使用 SSL。)

为了理解 SSL 的需求，我们浏览一下某典型的因特网商业的场景。Bob 在 Web 上冲浪，到达了 Alice 公司的站点，这个站点正在出售香水。Alice 公司站点显示了一个表格，假定 Bob 可以在该表格中输入香水的类型和所希望的数量、他的地址和他的支付卡号等信息。Bob 输入这些信息，点击“提交”，就期待收到(通过普通邮政邮件)所购买的香水；他也期待着在他的下一次支付卡报表中收到对所购物品的支付信息。所有这一切听起来不错，但是如果不采取安全措施，Bob 也许会有一些意外。

- 如果没有使用机密性(加密)，一个入侵者可能截取 Bob 的订单并得到他的支付卡信息。这个入侵者则可以用 Bob 的费用来购买商品。
- 如果没有使用完整性，入侵者可能修改 Bob 的订单，让他购买比希望瓶数多 10 倍的香水。
- 最后，如果没有使用服务器鉴别，这个显示 Alice 公司著名徽标的服务器实际上是由 Trudy 维护的一个站点，Trudy 正在假冒 Alice 公司。当 Trudy 收到 Bob 的订单后，可能拿了 Bob 的钱一走了之。或者 Trudy 可能充当一名身份窃贼，收集 Bob 的名字、地址和信用卡号。

SSL 通过采用机密性、数据完整性、服务器鉴别和客户鉴别来强化 TCP，就可以解决这些问题。

SSL 经常用来为发生在 HTTP 之上的事务提供安全性。然而，因为 SSL 使 TCP 安全了，因此它能被应用于运行在 TCP 之上的任何应用程序。SSL 提供了一个简单的具有套接字的应用编程接口(API)，该接口类似于 TCP 的 API。当一个应用程序要使用 SSL 时，它包括了 SSL 类/库。如在图 8-24 中所示，尽管 SSL 技术上位于应用层中，但从研发者的角度看，它是一个提供 TCP 服务的运输协议，而这里的 TCP 服务用安全性服务加强了。

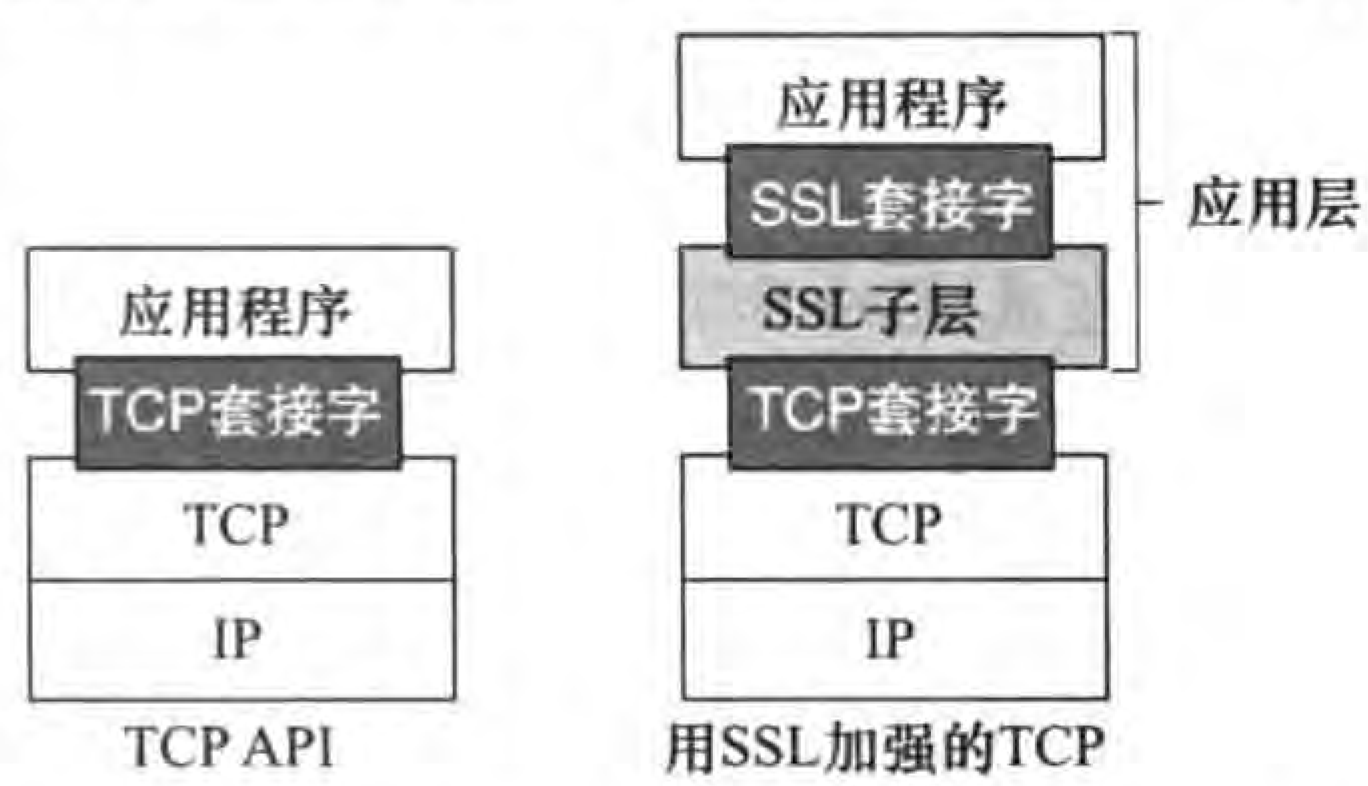


图 8-24 尽管从 SSL 技术上看位于应用层中，但从研发者的角度看它是一个运输层协议

8.6.1 宏观描述

我们从描述一个简化的 SSL 版本开始，这将使我们从宏观上理解 SSL 的工作原理和工作过程。我们将这个 SSL 的简化版本称之为“类 SSL”。描述过类 SSL 之后，在下一小节中我们将描述真实的 SSL，填充细节。类 SSL(和 SSL)具有三个阶段：握手、密钥导出和数据传输。我们现在描述针对一个客户(Bob)和一个服务器(Alice)之间的通信会话的这三个阶段，其中 Alice 具有私钥/公钥对和将她的身份与其公钥绑定的证书。

1. 握手

在握手阶段，Bob 需要：①与 Alice 创建一条 TCP 连接；②验证 Alice 是真实的 Alice；③发送给 Alice 一个主密钥，Bob 和 Alice 持用该主密钥生成 SSL 会话所需的所有对称密

钥。这三个步骤显示在图 8-25 中。注意到一旦创建了 TCP 连接，Bob 就向 Alice 发送一个 hello 报文。Alice 则用她的证书进行响应，证书中包含了她的公钥。如在 8.3 节所讨论，因为该证书已被某 CA 证实过，Bob 明白无误地知道该公钥属于 Alice。然后，Bob 产生一个主密钥（MS）（该 MS 将仅用于这个 SSL 会话），用 Alice 的公钥加密该 MS 以生成加密的主密钥（EMS），并将该 EMS 发送给 Alice。Alice 用她的私钥解密该 EMS 从而得到该 MS。在这个阶段后，Bob 和 Alice（而无别的人）均知道了用于这次 SSL 会话的主密钥。

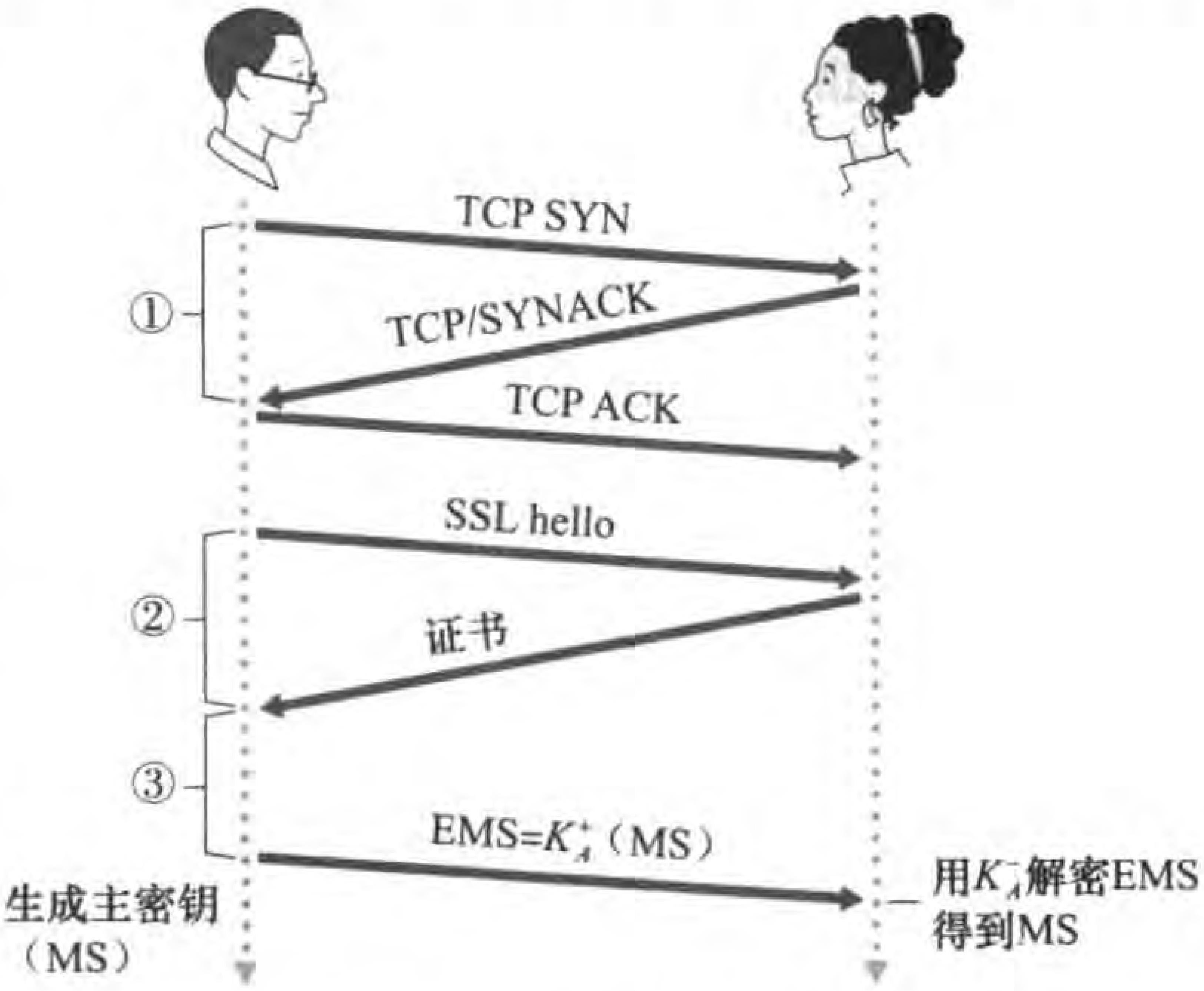


图 8-25 类 SSL 握手，首先建立一个 TCP 连接

2. 密钥导出

从原则上讲，MS 此时已由 Bob 和 Alice 共享，它能够用作所有后继加密和数据完整性检查的对称会话密钥。然而，对于 Alice 和 Bob 每人而言，使用不同的密码密钥，并且对于加密和完整性检查也使用不同的密钥，通常认为更为安全。因此，Alice 和 Bob 都使用 MS 生成 4 个密钥：

- E_B，用于从 Bob 发送到 Alice 的数据的会话加密密钥
- M_B，用于从 Bob 发送到 Alice 的数据的会话 MAC 密钥
- E_A，用于从 Alice 发送到 Bob 的数据的会话加密密钥
- M_A，用于从 Alice 发送到 Bob 的数据的会话 MAC 密钥

Alice 和 Bob 每人都从 MS 生成 4 个密钥。这能够通过直接将该 MS 分为 4 个密钥来实现。（但在真实的 SSL 中更为复杂一些，我们后面将会看到。）在密钥导出阶段结束时，Alice 和 Bob 都有了 4 个密钥。其中的两个加密密钥将用于加密数据；两个 MAC 密钥将用于验证数据的完整性。

3. 数据传输

既然 Alice 和 Bob 共享相同的 4 个会话密钥（E_B，M_B，E_A 和 M_A），他们就能够经 TCP 连接开始发送安全的数据。因为 TCP 是一种字节流协议，一种自然的方法是用 SSL 在传输中加密应用数据，然后将加密的数据在传输中传给 TCP。但是如果我们真的这样做，我们将用于完整性检查的 MAC 置于何处呢？我们无疑不希望等到 TCP 会话结束时才验证所有 Bob 数据的完整性，Bob 数据的发送要经历整个会话！为了解决这个问题，SSL 将数据流分割成记录，对每个记录附加一个 MAC 用于完整性检查，然后加密该“记录 + MAC”。为了产生这个 MAC，Bob 将数据连同密钥 M_B 放入一个散列函数中，如在 8.3 节所讨论。为了加密“记录 + MAC”这个包，Bob 使用他的会话加密密钥 E_B。然后这个加密的包将传递给 TCP 经因特网传输。

虽然这种方法几经周折，但它为整个报文流提供数据完整性时仍未达到无懈可击。特别是，假定 Trudy 是一名“中间人”，并且有在 Alice 和 Bob 之间发送的 TCP 报文段流中插入、删除和代替报文段的能力。例如，Trudy 能够俘获由 Bob 发送的两个报文段，颠倒这两个报文段的次序，调整 TCP 报文段的序号（这些未被加密），然后将这两个次序翻转的

报文段发送给 Alice。假定每个 TCP 报文段正好封装了一个记录，我们现在看看 Alice 将如何处理这些报文段。

1）在 Alice 端运行的 TCP 将认为一切正常，将这两个记录传递给 SSL 子层。

2）在 Alice 端的 SSL 将解密这两个记录。

3）在 Alice 端的 SSL 将使用在每个记录中的 MAC 来验证这两个记录的数据完整性。

4）然后 SSL 将解密的两条记录的字节流传递给应用层；但是 Alice 收到的完整字节流由于记录的颠倒而次序不正确！

鼓励读者观察类似的场景，如当 Trudy 删除报文段或当 Trudy 重放报文段时。

对该问题的解决方案如你可能猜想的那样，那就是使用序号。SSL 采用如下的方式。Bob 维护一个序号计数器，计数器开始为 0，Bob 每发送的一个 SSL 记录它都增加 1。Bob 并不实际在记录中包括一个序号，但当他计算 MAC 时，他把该序号包括在 MAC 的计算中。所以，该 MAC 现在是数据加 MAC 密钥 M_B 加当前序号的散列。Alice 跟踪 Bob 的序号，通过在 MAC 的计算中包括适当的序号，使她验证一条记录的数据完整性。SSL 序号的使用阻止了 Trudy 执行诸如重排序或重放报文段等中间人攻击。（为什么?）

4. SSL 记录

SSL 记录（以及类 SSL 记录）显示在图 8-26 中。该记录由类型字段、版本字段、长度字段、数据字段和 MAC 字段组成。注意到前三个字段是不加密的。类型字段指出了该字段是握手报文还是包含应用数据的报文。它也用于关闭 SSL 连接，如下面所讨论。在接收端的 SSL 使用长度字段以从到达的 TCP 字节流中提取 SSL 记录。版本字段是自解释的。

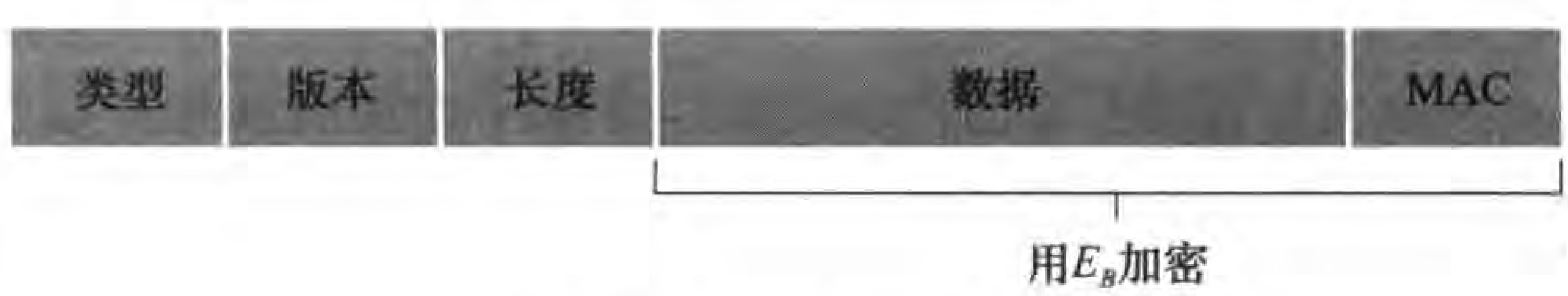


图 8-26　SSL 记录格式

8.6.2　更完整的描述

前一小节涉及了类 SSL 协议；其目的是让我们对 SSL 的工作原理和工作过程有一个基本理解。既然我们已经对 SSL 有了基本了解，就能够更深入地研究实际 SSL 协议的要点了。为了配合阅读对 SSL 协议的描述，鼓励读者完成 Wireshark SSL 实验，它在本书配套的 Web 网站上可供使用。

1. SSL 握手

SSL 并不强制 Alice 和 Bob 使用一种特定的对称密钥算法、一种特定的公钥算法或一种特定的 MAC。相反，SSL 允许 Alice 和 Bob 在握手阶段在 SSL 会话开始时就密码算法取得一致。此外，在握手阶段，Alice 和 Bob 彼此发送不重数，该数被用于会话密钥（E_B，M_B，E_A 和 M_A）的生成中。真正的 SSL 握手的步骤如下：

1）客户发送它支持的密码算法的列表，连同一个客户的不重数。

2）从该列表中，服务器选择一种对称算法（例如 AES）、一种公钥算法（例如具有特定密钥长度的 RSA）和一种 MAC 算法。它把它的选择以及证书和一个服务器不重数返回给客户。

3）客户验证该证书，提取服务器的公钥，生成一个**前主密钥**（Pre-Master Secret，PMS），用服务器的公钥加密该PMS，并将加密的PMS发送给服务器。

4）使用相同的密钥导出函数（就像SSL标准定义的那样），客户和服务器独立地从PMS和不重数中计算出**主密钥**（Master Secret，MS）。然后该MS被切片以生成两个密码和两个MAC密钥。此外，当选择的对称密码应用于CBC（例如3DES或AES），则两个初始化向量（Initialization Vector，IV）也从该MS获得，这两个IV分别用于该连接的两端。自此以后，客户和服务器之间发送的所有报文均被加密和鉴别（使用MAC）。

5）客户发送所有握手报文的一个MAC。

6）服务器发送所有握手报文的一个MAC。

最后两个步骤使握手免受篡改危害。为了理解这一点，观察在第一步中，客户通常提供一个算法列表，其中有些算法强，有些算法弱。因为这些加密算法和密钥还没有被协商好，所以算法的这张列表以明文形式发送。Trudy作为中间人，能够从列表中删除较强的算法，迫使客户选择一种较弱的算法。为了防止这种篡改攻击，在步骤5中客户发送一个级联它已发送和接收的所有握手报文的MAC。服务器能够比较这个MAC与它已接收和发送的握手报文的MAC。如果有不一致，服务器能够终止该连接。类似地，服务器发送一个它已经看到的握手报文的MAC，允许客户检查不一致性。

你可能想知道在步骤1和步骤2中存在不重数的原因。序号不足以防止报文段重放攻击吗？答案是肯定的，但它们并不只是防止“连接重放攻击”。考虑下列连接重放攻击。假设Trudy嗅探了Alice和Bob之间的所有报文。第二天，Trudy冒充Bob并向Alice发送正好是前一天Bob向Alice发送的相同的报文序列。如果Alice没有使用不重数，她将以前一天发送的完全相同的序列报文进行响应。Alice将不怀疑任何不规矩的事，因为她接收到的每个报文将通过完整性检查。如果Alice是一个电子商务服务器，她将认为Bob正在进行第二次订购（正好订购相同的东西）。在另一方面，在协议中包括了一个不重数，Alice将对每个TCP会话发送不同的不重数，使得这两天的加密密钥不同。因此，当Alice接收到来自Trudy重放的SSL记录时，该记录将无法通过完整性检查，并且假冒的电子商务事务将不会成功。总而言之，在SSL中，不重数用于防御“连接重放”，而序号用于防御在一个进行中的会话中重放个别分组。

2. 连接关闭

在某个时刻，Bob或者Alice将要终止SSL会话。一个方法是让Bob通过直接终止底层的TCP连接来结束该SSL会话，这就是说，通过让Bob向Alice发送一个TCP FIN报文段。但是这种幼稚设计为*截断攻击*（truncation attack）创造了条件，Trudy再一次介入一个进行中的SSL会话中，并用TCP FIN过早地结束了该会话。如果Trudy这样做的话，Alice将会认为她收到了Bob的所有数据，而实际上她仅收到了其中的一部分。对这个问题的解决方法是，在类型字段中指出该记录是否是用于终止该SSL会话的。（尽管SSL类型是以明文形式发送的，但在接收方使用了记录的MAC对它进行了鉴别。）通过包括这样一个字段，如果Alice在收到一个关闭SSL记录之前突然收到了一个TCP FIN，她可能知道正在进行着某些要花招的事情。

到此为止完成了对SSL的介绍。我们已经看到它使用了在8.2节和8.3节讨论的许多密码学原则。希望更深入地探讨SSL的读者可以阅读Rescorla的有关SSL的可读性很强的书籍［Rescorla 2001］。

8.7 网络层安全性：IPsec 和虚拟专用网

IP 安全（IP Security）协议更常被称为 **IPsec**，它为网络层提供了安全性。IPsec 为任意两个网络层实体（包括主机和路由器）之间的 IP 数据报提供了安全。如我们很快要描述的那样，许多机构（公司、政府部门、非营利组织等等）使用 IPsec 创建了运行在公共因特网之上的**虚拟专用网**（Virtual Private Network，VPN）。

在学习 IPsec 细节之前，我们后退一步来考虑为网络层提供机密性所包含的意义。在网络实体对之间（例如，两台路由器之间，两台主机之间，或者路由器和主机之间）具有网络层机密性，发送实体加密它发送给接收实体的所有数据报的载荷。这种载荷可以是一个 TCP 报文段、一个 UDP 报文段、一个 ICMP 报文等等。如果这样的网络层服务适当的话，从一个实体向其他实体发送的所有数据报将隐形于任何可能嗅探该网络的第三方，发送的数据报包括电子邮件、Web 网页、TCP 握手报文和管理报文（例如 ICMP 和 SNMP）。正因为如此，网络层安全性被认为提供了“地毯覆盖”。

除了机密性，网络层安全协议潜在地能够提供其他安全性服务。例如，它能提供源鉴别，使得接收实体能够验证安全数据报的源。网络层安全协议能够提供数据完整性，使得接收实体能够核对在数据报传输过程中可能出现的任何篡改。网络层安全服务也能提供防止重放攻击功能，这意味着 Bob 能够检测任何攻击者可能插入的任何冗余数据报。我们将很快看到 IPsec 的确提供了用于这些安全服务的机制，即机密性、源鉴别、数据完整性和重放攻击防护。

8.7.1 IPsec 和虚拟专用网

跨越在多个地理区域上的某机构常常希望有自己的 IP 网络，使它的主机和服务器能够以一种安全和机密的方式彼此发送数据。为了达到这个目标，该机构能够实际部署一个单独的物理网络，该网络包括路由器、链路和 DNS 基础设施且与公共因特网完全分离。这样一种为特定的机构专用的分立网络被称为**专用网络**（private network）。毫不奇怪，专用网络可能耗资巨大，因为该机构需要购买、安装和维护它自己的物理网络基础设施。

不同于部署和维护一个专用网络，如今许多机构在现有的公共因特网上创建 VPN。使用 VPN，机构办公室之间的流量经公共因特网而不是经物理上独立的网络发送。而为了提供机密性，办公室之间的流量在进入公共因特网之前进行加密。图 8-27 中显示了 VPN 的一个简单例子。这里的机构由一个总部、一个分支机构和旅行中的销售员组成，销售员通常从他们的旅馆房间接入因特网。（在该图中仅显示了一名销售员。）在这个 VPN 中，无论何时，位于总部的两台主机相互发送 IP 数据报或位于分支机构的两台主机要通信，它们都使用经典的 IPv4（即无 IPsec 服务）。然而，当两台机构的主机经过跨越公共因特网的路径时，这些流量在进入因特网之前进行加密。

为了感受 VPN 的工作过程，我们浏览图 8-27 场景中的一个简单例子。当总部中的一台主机向某旅馆中的某销售员发送一个 IP 数据报时，总部中的网关路由器将经典的 IPv4 转换成为 IPsec 数据报，然后将该 IPsec 数据报转发进因特网。该 IPsec 数据报实际上具有传统的 IPv4 首部，因此在公共因特网中的路由器处理该数据报，仿佛它对路由器而言是一个普通的 IPv4 数据报。但是如图 8-27 所示，IPsec 数据报的载荷包括了一个 IPsec 首部，该首部被用于 IPsec 处理；此外，IPsec 数据报的载荷是被加密的。当该 IPsec 数据报到达

销售员的便携机时，便携机的操作系统解密载荷（并提供其他安全服务，如验证数据完整性），并将解密的载荷传递给上层协议（例如，给 TCP 或 UDP）。

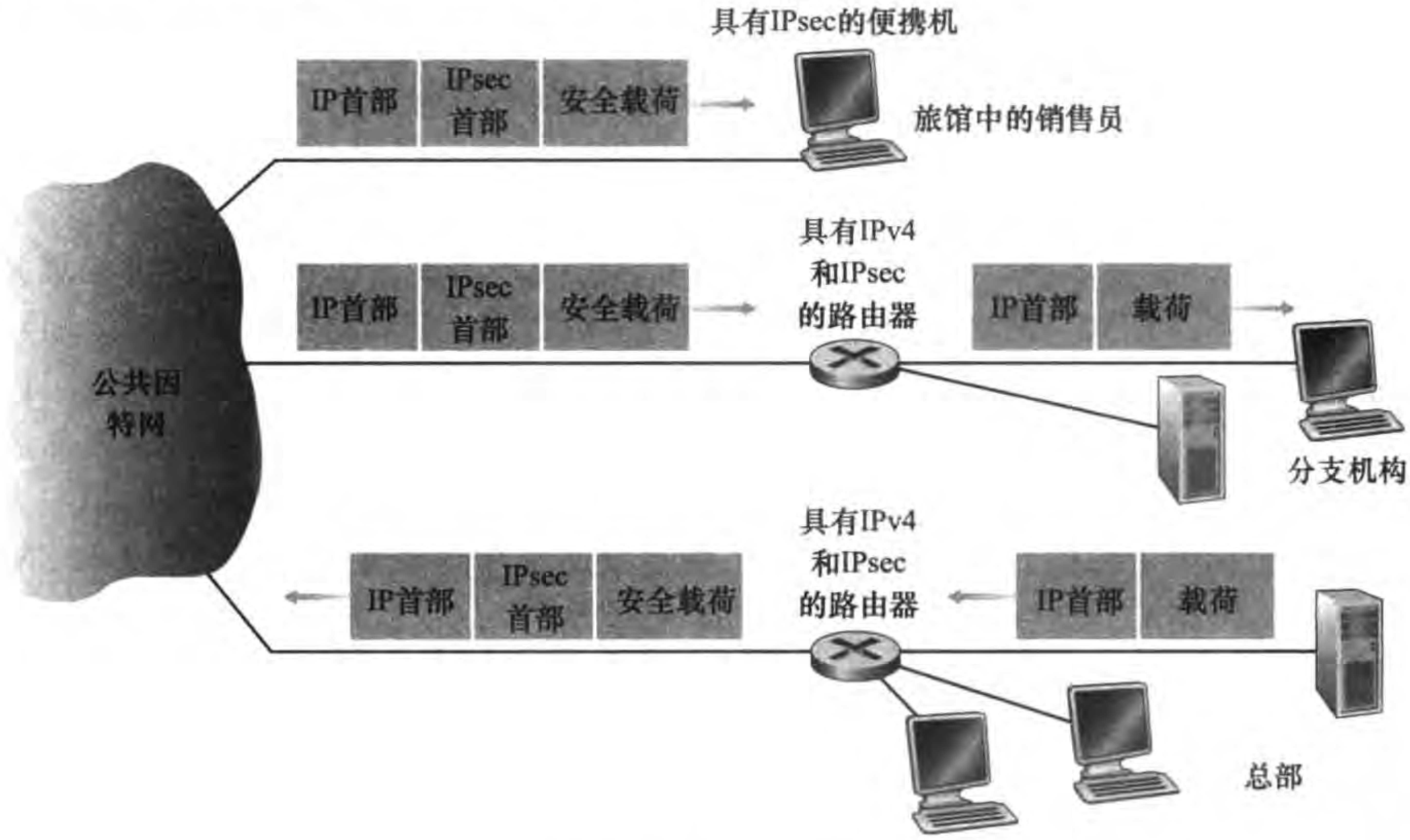


图 8-27　虚拟专用网

我们刚刚给出了某机构能够应用 IPsec 生成一个 VPN 的高层面的展望。为了通过局部看全局，我们已经去掉了许多重要的细节。现在我们来进行更深入的学习。

8.7.2 AH 协议和 ESP 协议

IPsec 是一个相当复杂的整体，即它被定义为 10 多个 RFC 文档。两个重要的文档是 RFC 4301 和 RFC 6071，前者描述了总体 IP 安全体系结构，后者提供了一个 IPsec 协议集的概述。在本教科书中我们的目标与往常一样，并不只是一味重复枯燥和晦涩难解的 RFC 文档，而是采用一种更具可操作性和易于教学的方法来描述协议。

在 IPsec 协议族中，有两个主要协议：**鉴别首部**（Authentication Header，AH）协议和**封装安全性载荷**（Encapsulation Security Payload，ESP）协议。当某源 IPsec 实体（通常是一台主机或路由器）向一个目的实体（通常也是一台主机或路由器）发送安全数据报时，它可以使用 AH 协议或 ESP 协议来做到。AH 协议提供源鉴别和数据完整性服务，但不提供机密性服务。ESP 协议提供了源鉴别、数据完整性和机密性服务。因为机密性通常对 VPN 和其他 IPsec 应用是至关重要的，所以 ESP 协议的使用比 AH 协议要广泛得多。为了讲清 IPsec 并且避免许多难题，我们将此后专门关注 ESP 协议。鼓励还想学习 AH 协议的读者研讨相关的 RFC 和其他在线资源。

8.7.3 安全关联

IPsec 数据报在网络实体对之间发送，例如两台主机之间、两台路由器之间或一台主机和一台路由器之间。在从源实体向目的实体发送 IPsec 数据报之前，源和目的实体创建了一个网络层的逻辑连接。这个逻辑连接称为**安全关联**（Security Association，SA）。一个 SA 是一个单工逻辑连接；也就是说，它是从源到目的地单向的。如果两个实体要互相发

送安全数据报，则需创建两个 SA，每个方向一个。

例如，再次考虑图 8-27 中那个机构的 VPN。该机构由一个总部、一个分支机构和 n 个旅行销售员组成。为了举例的缘故，我们假设在总部和分支机构之间有双向 IPsec 流量，并且总部和销售员之间也有双向 IPsec 流量。在这个 VPN 中，有多少个 SA 呢？为了回答这个问题，注意到在总部网关路由器和分支机构网关路由器之间有两个 SA（一个方向一个）；对每个销售员的便携机而言，在总部网关和便携机之间有两个 SA（仍是一个方向一个）。因此，总计为 $(2+2n)$ 个 SA。*然而记住，并非从网关路由器或便携机发送进因特网的所有流量都将是 IPsec 安全的*。例如，总部中的一台主机可能要访问公共因特网中的某 Web 服务器（例如 Amazon 或谷歌）。因此，该网关路由器（或该便携机）将发送经典的 IPv4 数据报和安全的 IPsec 数据报进入因特网。

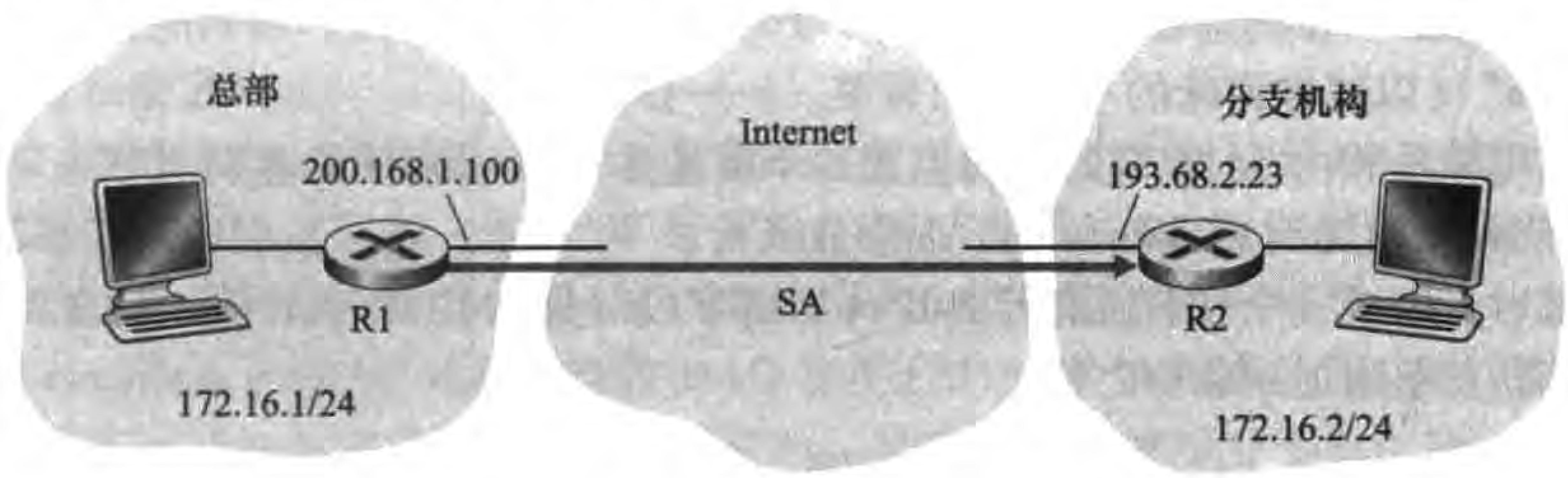


图 8-28 从 R1 到 R2 的安全关联

我们现在观察 SA 的"内部"。为了使讨论明确和具体，我们在图 8-28 中的一个从路由器 R1 到路由器 R2 的 SA 场景下进行观察。（你能够认为路由器 R1 是图 8-27 中的总部网关路由器，而路由器 R2 是图 8-27 中的分支机构网关路由器。）路由器 R1 将维护有关该 SA 的状态信息，这将包括：

- SA 的 32 比特的标识符，称为**安全参数索引**（Security Parameter Index，SPI）。
- SA 的初始接口（在此例中为 200.168.1.100）和 SA 的目的接口（在此例中为 193.68.2.23）。
- 将使用的加密类型（例如，具有 CBC 的 3DES）。
- 加密密钥。
- 完整性检查的类型（例如，具有 MD5 的 HMAC）。
- 鉴别密钥。

无论何时路由器 R1 需要构建一个 IPsec 数据报经过这个 SA 转发，它访问该状态信息以决定它应当如何鉴别和加密该数据报。类似地，路由器 R2 将维护对此 SA 的相同的状态信息，并将使用该信息鉴别和加密任何从该 SA 到达的 IPsec 数据报。

一个 IPsec 实体（路由器或主机）经常维护许多 SA 的状态信息。例如，在图 8-27 中具有 n 个销售员的 VPN 例子中，总部网关路由器维护 $(2+2n)$ 个 SA 的状态信息。一个 IPsec 实体在它的**安全关联数据库**（Security Association Database，SAD）中存储其所有 SA 的状态信息，SAD 是实体操作系统内核中的一个数据结构。

8.7.4 IPsec 数据报

在描述了 SA 后，我们现在能够描述实际的 IPsec 数据报了。IPsec 有两种不同的分组形式，一种用于所谓**隧道模式**（tunnel mode），另一种用于所谓**运输模式**（transport

mode)。更为适合 VPN 的隧道模式比运输模式部署得更为广泛。为了进一步讲清 IPsec 和避免许多难题，我们因此专门关注隧道模式。一旦已经牢牢地掌握了隧道模式，应当能够容易地自学运输模式。

IPsec 数据报的分组格式显示在图 8-29 中。你也许认为分组格式是枯燥乏味的，但我们将很快看到 IPsec 数据报实际上尝起来像美式墨西哥风味（Tex-Mex）美食！我们考察图 8-28 的场景中的 IPsec 字段。假设路由器 R1 接收到一个来自主机 172.16.1.17（在总部网络中）的普通 IPv4 数据报，该分组的目的地是主机 172.16.2.48（在分支机构网络中）。路由器 R1 使用下列方法将这个“普通 IPv4 数据报”转换成一个 IPsec 数据报：

- 在初始 IPv4 数据报（它包括初始首部字段！）后面附上一个“ESP 尾部”字段。
- 使用算法和由 SA 规定的密钥加密该结果。
- 在这个加密量的前面附加上一个称为“ESP 首部”的字段，得到的包称为“enchilada”（以辣椒调味的一种墨西哥菜。——译者注）。
- 使用算法和由 SA 规定的密钥生成一个覆盖整个 enchilada 的鉴别 MAC。
- 该 MAC 附加到 enchilada 的后面形成载荷。
- 最后，生成一个具有所有经典 IPv4 首部字段（通常共 20 字节长）的全新 IP 首部，该新首部附加到载荷之前。

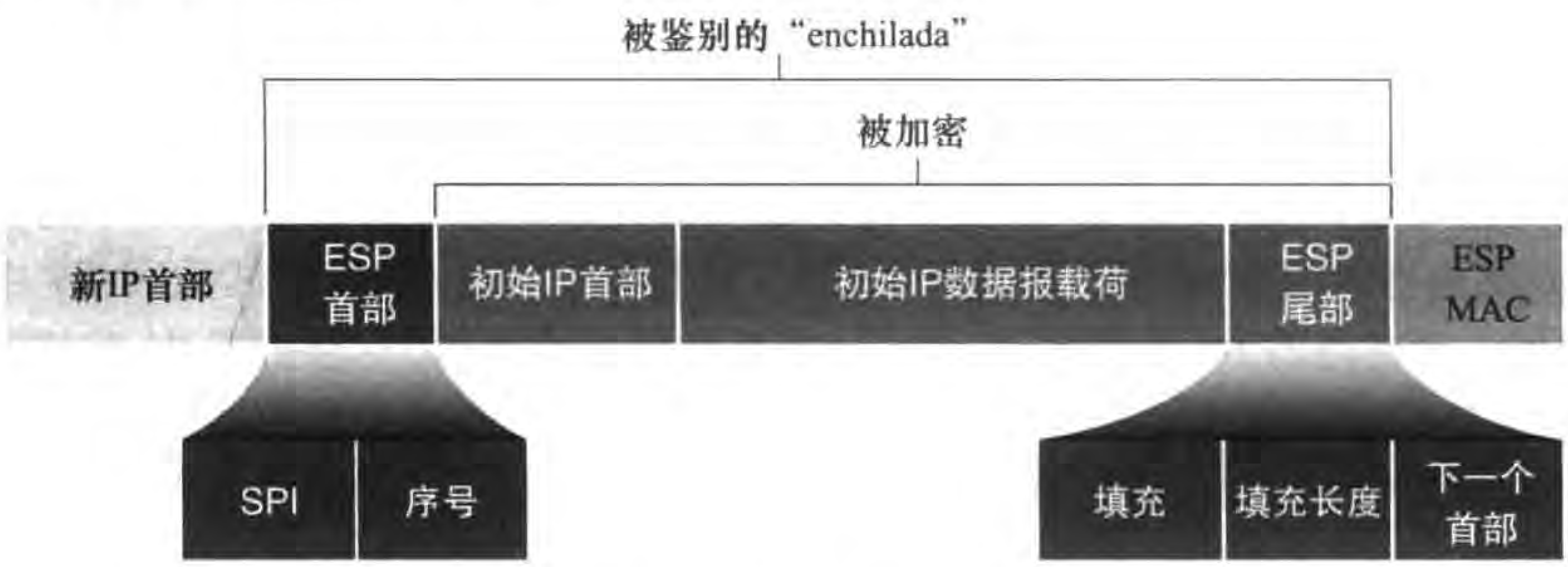


图 8-29　IPsec 数据报格式

注意到得到的 IPsec 数据报是一个货真价实的 IPv4 数据报，它具有传统的 IPv4 首部字段后跟一个载荷。但在这个场合，该载荷包含一个 ESP 首部、初始 IP 数据报、一个 ESP 尾部和一个 ESP 鉴别字段（具有加密的初始数据报和 ESP 尾部）。初始的 IP 数据报具有源 IP 地址 172.16.1.17 和目的地址 172.16.2.48。因为 IPsec 数据报包括了该初始 IP 数据报，这些地址被包含和被加密作为 IPsec 分组负载的组成部分。但是在新 IP 首部中的源和目的地 IP 地址，即在 IPsec 数据报的最左侧首部又该如何处理呢？如你所猜测，它们被设置为位于隧道两个端点的源和目的地路由器接口，也就是 200.168.1.100 和 193.68.2.23。同时，这个新 IPv4 首部字段中的协议号不被设置为 TCP、UDP 或 SMTP，而是设置为 50，指示这是一个使用 ESP 协议的 IPsec 数据报。

在 R1 将 IPsec 数据报发送进公共因特网之后，它在到达 R2 之前将通过许多路由器。这些路由器中的每个将处理该数据报，就像它是一个普通数据报一样，即它们被完全忘记这样的事实：该数据报正在承载 IPsec 加密的数据。对于这些公共因特网路由器，因为在外面首部中的目的 IP 地址是 R2，所以该数据报的最终目的地是 R2。

在考察了如何构造一个 IPsec 数据报的例子后，我们现在更仔细地观察 enchilada 的组成。我们看到在图 8-29 中的 ESP 尾部由三个字段组成：填充、填充长度和下一个首部。

前面讲过块密码要求被加密的报文必须为块长度的整数倍。使用填充（由无意义的字节组成），使得当其加上初始数据报（连同填充长度字段和下一个首部字段）形成的“报文”是块的整数倍。填充长度字段指示接收实体插入的填充是多少（并且需要被删除）。下一个首部字段指示包含在载荷数据字段中数据的类型（例如 UDP）。载荷数据（通常是初始 IP 数据报）和 ESP 尾部级联起来并被加密。

附加到这个加密单元前面的是 ESP 首部，该首部以明文发送，它由两个字段组成：SPI 字段和序号字段。SPI 字段指示接收实体该数据报属于哪个 SA；接收实体则能够用该 SPI 索引其 SAD 以确定适当的鉴别/解密算法和密钥。序号字段用于防御重放攻击。

发送实体也附加一个鉴别 MAC。如前所述，发送实体跨越整个 enchilada（由 ESP 首部、初始 IP 数据报和 ESP 尾部组成，即具有加密的数据报和尾部）计算一个 MAC。前面讲过为了计算一个 MAC，发送方附加一个秘密 MAC 密钥到该 enchilada，进而计算该结果的一个固定长度散列。

当 R2 接收到 IPsec 数据报时，R2 看到该数据报的目的 IP 地址是 R2 自身。R2 因此处理该数据报。因为协议字段（位于 IP 首部最左侧）是 50，R2 明白应当对该数据报施加 IPsec ESP 处理。第一，针对 enchilada，R2 使用 SPI 以确定该数据报属于哪个 SA。第二，它计算该 enchilada 的 MAC 并且验证该 MAC 与在 ESP MAC 字段中的值一致。如果两者一致，它知道该 enchilada 来自 R1 并且未被篡改。第三，它检查序号字段以验证该数据报是新的（并且不是重放的数据报）。第四，它使用与 SA 关联的解密算法和密钥解密该加密单元。第五，它删除填充并抽取初始的普通 IP 报文。最后，它朝着其最终目的地将该初始数据报转发进分支机构网络。这个一种多么复杂的秘诀呀！还未曾有人声称准备并破解 enchilada 是一件容易的事！

实际上还有另一个重要的细微差别需要处理。它以下列问题为中心：当 R1 从位于总部网络中的一台主机收到一个（未加密的）数据报时，并且该数据报目的地为总部以外的某个目的 IP 地址，R2 怎样才能知道它应当将其转换为一个 IPsec 数据报呢？并且如果它由 IPsec 处理，R1 如何知道它应当使用（在其 SAD 中的许多 SA 中）哪个 SA 来构造这个 IPsec 数据报呢？该问题以如下方式解决。除了 SAD 外，IPsec 实体也维护另一个数据结构，它称为**安全策略库**（Security Policy Database，SPD）。该 SPD 指示哪些类型的数据报（作为源 IP 地址、目的 IP 地址和协议类型的函数）将被 IPsec 处理；并且对这些将被 IPsec 处理的数据报应当使用哪个 SA。从某种意义上讲，在 SPD 中的信息指示对于一个到达的数据报做“什么”；在 SAD 中的信息指示“怎样”去做。

IPsec 服务的小结

IPsec 究竟提供什么样的服务呢？我们从某攻击者 Trudy 的角度来考察这些服务，Trudy 是一个中间人，位于图 8-28 中 R1 和 R2 之间路径上的某处。假设通过这些讨论，Trudy 不知道 SA 所使用的鉴别和加密密钥。Trudy 能够做些什么和不能够做些什么呢？第一，Trudy 不能看到初始数据报。如果事实如此，不仅 Trudy 看不到在初始数据报中的数据，而且也看不到协议号、源 IP 地址和目的 IP 地址。对于经该 SA 发送的数据报，Trudy 仅知道该数据报源于 172. 16. 1. 0/24 的某台主机以及目的地为 172. 16. 2. 0/24 的某台主机。她不知道它是否携带 TCP、UDP 或 ICMP 数据；她不知道它是否携带了 HTTP、SMTP 或某些其他类型的应用程序数据。因此这种机密性比 SSL 范围更为宽广。第二，Trudy 试图用反转数据报的某些比特来篡改在 SA 中的某个数据报，当该篡改的数据报到达 R2 时，它

将难以通过完整性核查（使用 MAC），再次挫败了 Trudy 的恶意尝试。第三，假设 Trudy 试图假冒 R1，生成一个源为 200.168.1.100 和目的地为 193.68.2.23 的 IPsec 数据报。Trudy 的攻击将是无效的，因为这个数据报将再次通不过在 R2 的完整性核查。最后，因为 IPsec 包含序号，Trudy 将不能够生成一个成功的重放攻击。总而言之，正如本节开始所言，IPsec 在任何通过网络层处理分组的设备对之间，提供了机密性、源鉴别、数据完整性和重放攻击防护。

8.7.5 IKE：IPsec 中的密钥管理

当某 VPN 具有少量的端点时（例如，图 8-28 中只有两台路由器），网络管理员能够在该端点的 SAD 中人工键入 SA 信息（加密/鉴别算法和密钥及 SPI）。这样的“人工密钥法”对于一个大型 VPN 显然是不切实际的，因为大型 VPN 可能由成百甚至上千台 IPsec 路由器和主机组成。大型的、地理上分散的部署要求一个自动的机制来生成 SA。IPsec 使用**因特网密钥交换**（Internet Key Exchange，IKE）协议来从事这项工作，IKE 由 RFC 5996 定义。

IKE 与 SSL（参见 8.6 节）中的握手具有某些类似。每个 IPsec 实体具有一个证书，该证书包括了该实体的公开密钥。如同使用 SSL 一样，IKE 协议让两个实体交换证书，协商鉴别和加密算法，并安全地交换用于在 IPsec SA 中生成会话密钥的密钥材料。与 SSL 不同的是，IKE 应用两个阶段来执行这些任务。

我们来研究图 8-28 中两台路由器 R1 和 R2 场景下的这两个阶段。第一个阶段由 R1 和 R2 之间报文对的两次交换组成：

- 在报文的第一次交换期间，两侧使用 Diffie-Hellman（参见课后习题）在路由器之间生成一个双向的 IKE SA。为了防止混淆，这个双向 IKE SA 完全不同于 8.6.3 节和 8.6.4 节所讨论的 IPsec SA。该 IKE SA 在这两台路由器之间提供了一个鉴别的和加密的信道。在首个报文对交换期间，创建用于 IKE SA 的加密和鉴别的密钥。还创建了将用于计算后期在阶段 2 使用的 IPsec SA 密钥的一个主密钥。观察在第一步骤期间，没有使用 RSA 公钥和私钥。特别是，R1 或 R2 都没有通过用它们的私钥对报文签字而泄露其身份。
- 在报文的第二次交换期间，两侧通过对其报文签名而透漏了它们的身份。然而，这些身份并未透漏给被动的嗅探者，因为这些报文是经过安全的 IKE SA 信道发送的。同时在这个阶段期间，两侧协商由 IPsec SA 应用的 IPsec 加密和鉴别算法。

在 IKE 的第二个阶段，两侧生成在每个方向的一个 SA。在阶段 2 结束时，对这两个 SA 的每一侧都建立了加密和鉴别会话密钥。然后这两侧都能使用 SA 来发送安全的数据报，如同 8.7.3 节和 8.7.4 节描述的那样。在 IKE 中有两个阶段的基本动机是计算成本，即因为第二阶段并不涉及任何公钥密码，IKE 能够以相对低的计算成本在两个 IPsec 实体之间生成大量 SA。

8.8 使无线 LAN 安全

在无线网络中安全性是特别重要的关注因素，因为这时携带数据帧的无线电波可以传播到远离包含无线基站和主机的建筑物以外的地方。在本节中，我们简要介绍了无线安全性。对于更为深入地探讨，参见由 Edney 和 Arbaugh 撰写的可读性很强的书 [Edney

2003]。

在802.11中的安全性问题受到了技术界和媒体界的极大关注。在进行大量讨论的同时，一个几乎没有争论的事实是，看起来被广泛认同的初始802.11规范具有一些严重的安全性缺陷。现在的确能够下载利用这些漏洞的公共域软件，使得那些使用该普通802.11安全性机制的用户面对安全性攻击，就像根本没有使用安全性措施的网络用户一样，门户洞开。

在下面一节中，我们讨论最初在802.11规范中标准化的安全性机制，该规范统称为**有线等效保密**（Wired Equivalent Privacy，WEP）。顾名思义，WEP意欲提供类似于在有线网络中的安全性水平。接下来我们将讨论WEP中的安全性漏洞并讨论802.11i标准，后者是在2004年采纳的802.11的本质上更为安全的版本。

8.8.1 有线等效保密

IEEE 802.11的WEP协议设计于1999年，为在主机和无线接入点（即基站）之间提供鉴别和数据的加密。WEP并没有指定密钥管理算法，因此它假定主机和无线接入点之间通过带外方式就密钥达成了某种一致。鉴别以下列方式进行：

1）无线主机通过接入点请求鉴别。

2）接入点以一个128字节的不重数值响应该鉴别请求。

3）无线主机用它与这个接入点共享的密钥加密这个不重数值。

4）接入点解密主机加密的不重数值。

如果解密所得不重数值与初始发给主机的值相同，则这个接入点鉴别了该主机。

图8-30阐述了WEP数据加密算法。假定主机和接入点都知道一个秘密的40比特对称密钥 K_S。此外，一个24比特的初始向量（IV）附加到这个40比特的密钥后面，产生用于加密单个帧的一个64比特密钥。每一个帧所使用的IV都不同，所以每一帧都由不同的64比特密钥加密。加密以如下方式进行。首先为每个数据载荷计算一个4字节的CRC值（见6.2节）。然后用RC4流密码加密该载荷和该4字节CRC。我们这里不涉及RC4的细节（细节参见［Schneier 1995］和［Edney 2003］）。就我们的目的而言，知道下列事实即可：对于密钥值（此时为64比特（K_S、IV）密钥），RC4算法产生一个密钥值的流为 k_1^{IV}，$k_2^{IV}, k_3^{IV}, \cdots$，这些密码值用于加密一帧中的数据和CRC值。出于实用的目的，我们可以认为每次对一个字节执行这些操作。通过把数据的第 i 字节 d_i 和由（K_S、IV）对生成的密钥值流中的第 i 个密钥 k_i^{IV} 执行异或操作进行加密，以产生密文的第 i 字节 c_i：

$$c_i = d_i \oplus k_i^{IV}$$

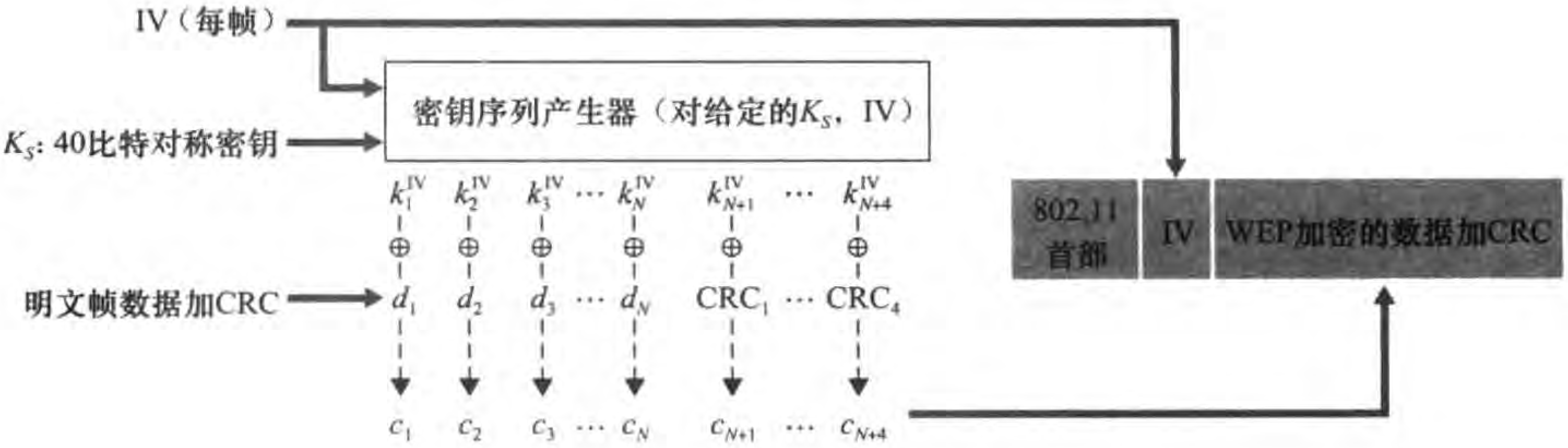


图8-30 802.11 WEP协议

如图 8-30 中所示，该 IV 值逐帧而变化，以明文形式出现在每一个 WEP 加密的 802.11 帧的首部中。接收方取它与发送方共享的秘密 40 比特对称密钥，添加上该 IV，并使用形成的 64 比特的密钥（它与发送方执行加密所用的密钥相同）来解密这个帧。

$$d_i = c_i \oplus k_i^{IV}$$

正确使用 RC4 算法要求同一个 64 比特密钥决不能使用超过 1 次。前面讲过 WEP 密钥是每一帧变换一次。对于某给定的 K_S（即使它有变化，也是很少的），这意味着只有 2^{24} 个不同的密钥可用。如果随机选择这些密钥的话，我们能够看到［Edney 2003］，则仅在处理 12 000 帧之后就选中相同的 IV 值（从而使用相同的 64 比特密钥）的概率超过 99%。在 1KB 帧长和 11Mbps 数据传输率的情况下，传输 12 000 帧仅需几秒的时间。此外，由于 IV 值在该帧中以明文形式传输，窃听者就会发现何时使用了一个重复的 IV 值。

为理解重复使用一个密钥可能出现的几个问题之一，考虑下面的选择明文攻击的情况，仍以 Trudy 对 Alice 进行攻击为例。假定 Trudy（可能使用 IP 哄骗）向 Alice 发出一个请求（例如，一个 HTTP 或 FTP 请求），要求 Alice 传输内容已知的文件 $d_1, d_2, d_3, d_4, \cdots$，Trduy 也观察到 Alice 发送的已加密数据 $c_1, c_2, c_3, c_4, \cdots$，由于 $d_i = c_i \oplus k_i^{IV}$，如果在这个等式两边同时异或 c_i，可得到：

$$d_i \oplus c_i = k_i^{IV}$$

根据这个关系，Trudy 就可以使用已知的 d_i 和 c_i 值计算出 k_i^{IV}。下一次 Trudy 看到使用同一 IV 值时，她将知道密钥流为 $k_1^{IV}, k_2^{IV}, k_3^{IV}, \cdots$，并可使用这些密钥解密报文。

对于 WEP 还有其他几个值得关注的安全性问题。［Fluhrer 2001］描述了一种攻击方法，即当选择某些弱密钥时在 RC4 中暴露出的一种已知弱点。［Stubblefield 2002］讨论了实现和开发这种攻击的有效方法。对 WEP 的另一种关注与在图 8-30 中显示并在 802.11 帧中传输的用以检测载荷中改变的比特的 CRC 比特有关。然而，攻击者在改变加密内容（例如用乱七八糟的东西替代初始的加密数据）后，对这些被替换的东西计算出一个 CRC，并将该 CRC 放置在 WEP 帧中产生一个将被接收方接受的 802.11 帧。此时所需要的是诸如我们在 8.3 节中学习的报文完整性技术来检测内容篡改或替换。有关 WEP 安全性更多的细节，参见［Edney 2003；Wright 2015］及其中的参考文献。

8.8.2　IEEE 802.11i

在 IEEE 802.11 于 1999 年发布后不久，就开始研发具有更强安全性机制的 802.11 的新型、改进版本。这个新标准被称为 802.11i，于 2004 年最终得到批准。如我们将看到的那样，虽然 WEP 提供了相对弱的加密、仅有单一方式执行鉴别并且没有密钥分发机制，但 IEEE 802.11i 却提供了强得多的加密形式、一种可扩展的鉴别机制集合以及一种密钥分发机制。下面我们概述一下 802.11i；［TechOnline 2012］是一篇关于 802.11i 的优秀（流式音频）技术概述。

图 8-31 概述了 802.11i 的框架。除了无线客户和接入点外，802.11i 定义了一台鉴别服务器，AP 能够与它通信。鉴别服务器与 AP 的分离使得一台鉴别服务器服务于许多 AP，集中在一台服务器中作出有关鉴别和接入（通常是敏感）的决定，降低了 AP 的成本和复杂性。802.11i 运行分为 4 个阶段：

1）发现。在发现阶段，AP 通告它的存在以及它能够向无线客户节点提供的鉴别和加密形式。客户则请求它希望的特定鉴别和加密形式。尽管客户和 AP 已经交换了报文，但

该客户还没有被鉴别，也还没有加密密钥，因此在该客户能够通过无线信道与任何远程主机通信之前，还需要进行几个其他步骤。

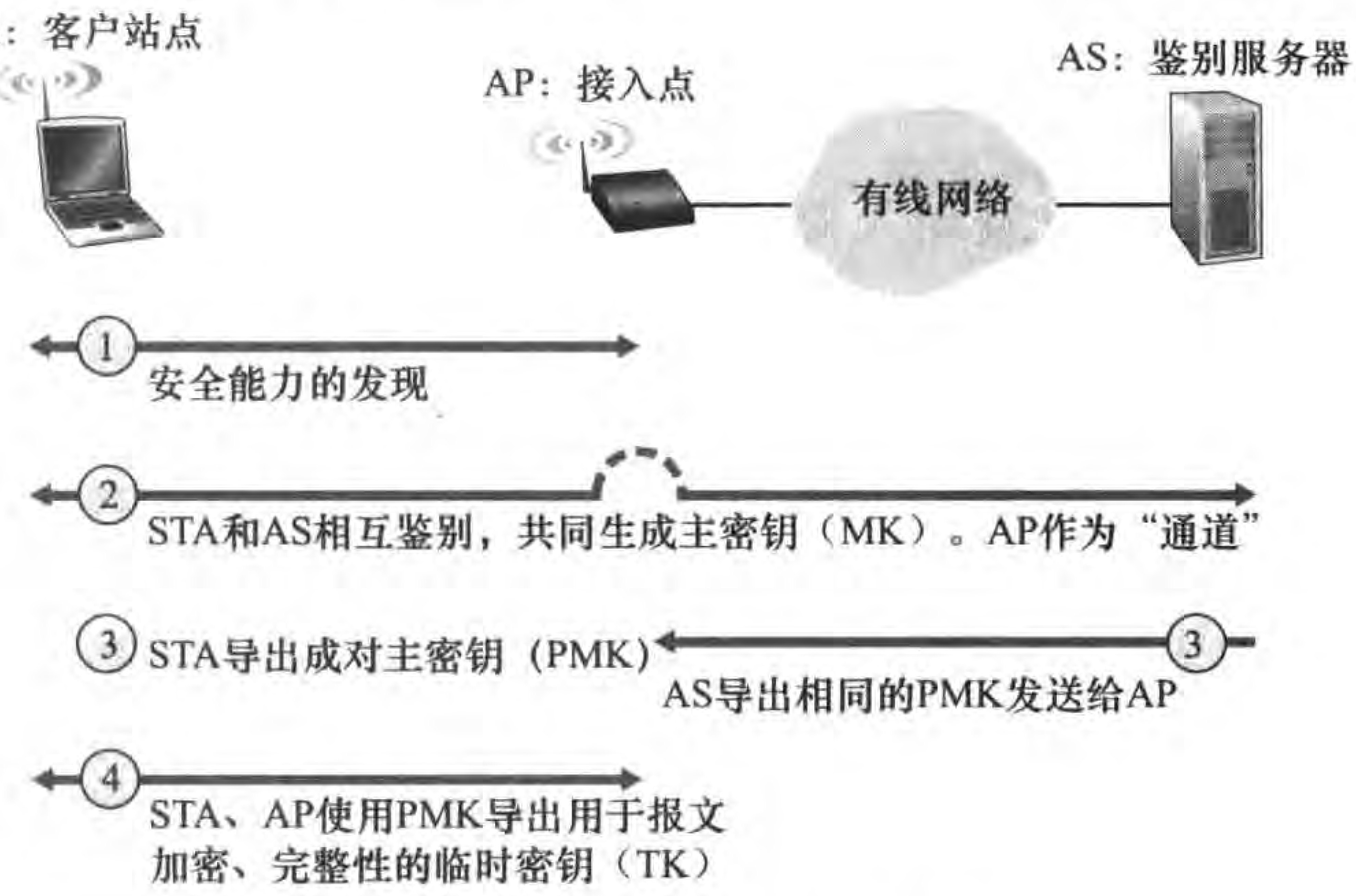


图 8-31 802.11i：运行的 4 个阶段

2）相互鉴别和主密钥（MK）生成。鉴别发生在无线客户和鉴别服务器之间。在这个阶段，接入点基本是起中继的作用，在客户和鉴别服务器之间转发报文。**可扩展鉴别协议**（Extensible Authentication Protocol，EAP）[RFC 3748] 定义了客户和鉴别服务器之间交互时简单的请求/响应模式中使用的端到端报文格式。如图 8-32 中所示，EAP 报文使用 **EAPoL**（EAP over LAN，[IEEE 802.1x]）进行封装，并通过 802.11 无线链路发送。这些 EAP 报文在接入点拆封，然后再使用 **RADIUS** 协议重新封装，经 UDP/IP 传输到鉴别服务器。尽管 RADIUS 服务器和协议 [RFC 2865] 并不为 802.11i 所要求，但它们是 802.11i 的事实上的标准组件。最近标准化的 **DIAMETER** 协议 [RFC 3588] 很可能在不久的将来替代 RADIUS。

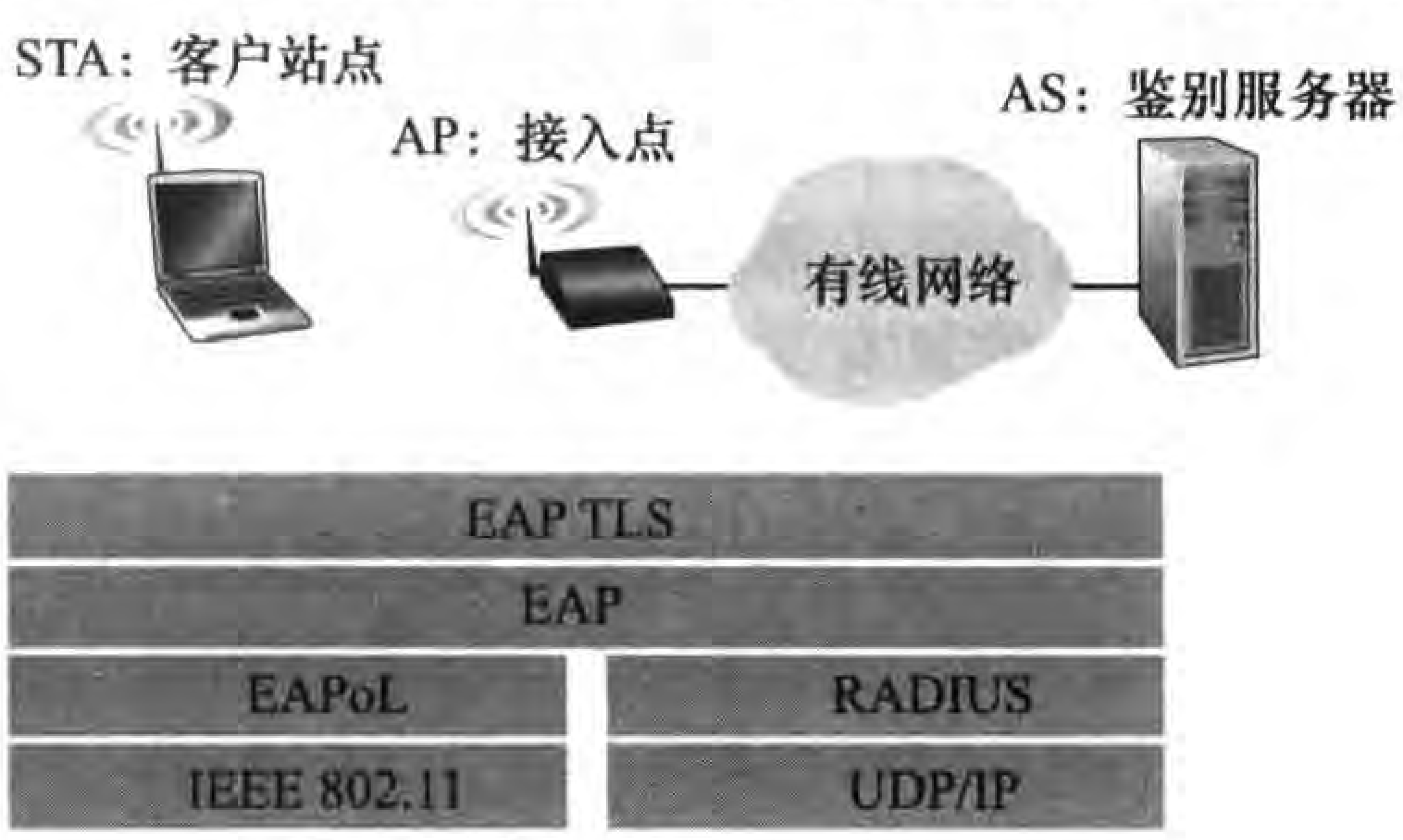


图 8-32 EAP 是一个端到端协议。EAP 报文使用 EAPoL（运行在客户和接入点之间的无线链路上）封装，并使用 RADIUS（运行在接入点和鉴别服务器之间的 UDP/IP 上）

使用 EAP，鉴别服务器能够选择若干方式中的一种来执行鉴别。802.11i 虽未强制一种特殊的鉴别方法，但经常使用 EAP-TLS 鉴别方案 [RFC 5216]。EAP-TLS 使用类似于我们在 8.3 节中研究的公钥技术（包括不重数加密和报文摘要），以允许客户和鉴别服务器彼此相互鉴别，并导出为双方所知的一个主密钥。

3）成对主密钥（Pairwise Master Key，PMK）生成。MK 是一个仅为客户和鉴别服务器所知的共享密钥，它们都使用 MK 来生成一个次密钥，即成对主密钥（PMK）。鉴别服务器则向 AP 发送该 PMK。这正是我们所希望达到的目的！客户和 AP 现在具有一个共享的密钥（前面讲过在 WEP 中根本不涉及密钥分发的问题），并彼此相互鉴别。它们此时已经快要能发挥效用了。

4）临时密钥（Temporal Key，TK）生成。使用 PMK，无线客户和 AP 现在能够生成

附加的、将用于通信的密钥。其中的关键是临时密钥，TK 将被用于执行经无线链路向任意远程主机发送数据的链路级的加密。

802.11i 提供了几种加密形式，其中包括基于 AES 的加密方案和 WEP 加密的强化版本。

8.9 运行安全性：防火墙和入侵检测系统

遍及本章我们已经看出，因特网不是一个很安全的地方，即有坏家伙出没，从事着各种各样的破坏活动。给定因特网的不利性质，我们现在考虑一个机构网络和管理它的网络管理员。从网络管理员的角度看，世界可以很明显地分为两个阵营：一部分是好人，他们属于本机构网络，可以用相对不受限制的方式访问该机构网络中的资源；另一部分是坏家伙，他们是其他一些人，访问网络资源时必须经过仔细审查。在许多机构中，从中世纪的城堡到现代公司的建筑物，都有单一的出口/入口，无论好人坏人出入该机构，都需要进行安全检查。在一个城堡中，可以在吊桥的一端的门口执行安全检查；在公司大厦中，这些工作可在安全台完成。在计算机网络中，当通信流量进入/离开网络时要执行安全检查、做记录、丢弃或转发，这些工作都由被称为防火墙、入侵检测系统（IDS）和入侵防止系统（IPS）的运行设备来完成。

8.9.1 防火墙

防火墙（firewall）是一个硬件和软件的结合体，它将一个机构的内部网络与整个因特网隔离开，允许一些数据分组通过而阻止另一些分组通过。防火墙允许网络管理员控制外部和被管理网络内部资源之间的访问，这种控制是通过管理流入和流出这些资源的流量实现的。防火墙具有3个目标：

- *从外部到内部和从内部到外部的所有流量都通过防火墙。*图8-33显示了一个防火墙，位于被管理网络和因特网其余部分之间的边界处。虽然许多大型机构可使用多级防火墙或分布式防火墙［Skoudis 2006］，但在对该网络的单一接入点处设置一个防火墙，如图8-33中所示，这使得管理和施加安全访问策略更为容易。

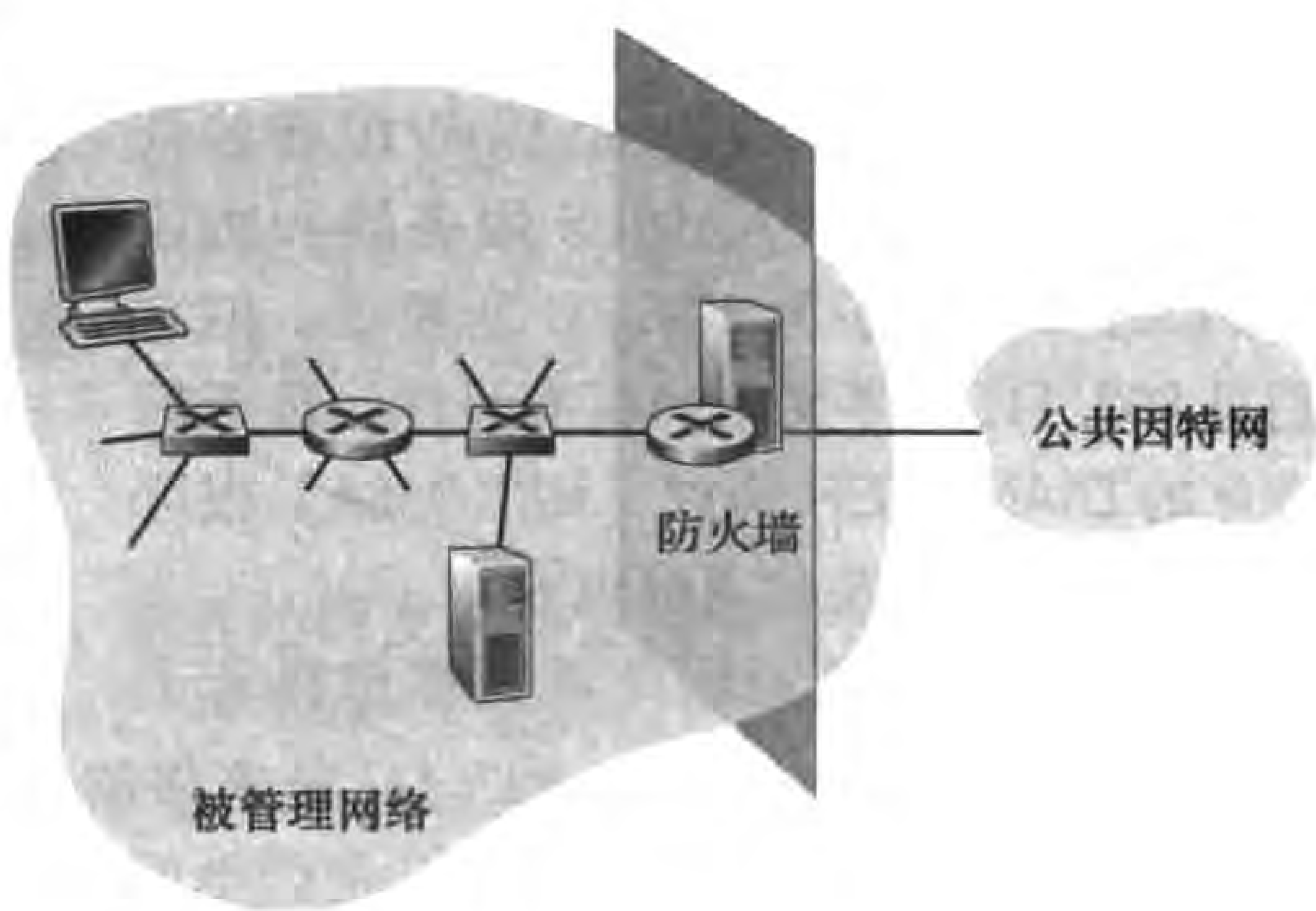


图8-33 在被管理网络和外部之间放置防火墙

- *仅被授权的流量（由本地安全策略定义）允许通过。*随着进入和离开机构网络的所有流量流经防火墙，该防火墙能够限制对授权流量的访问。
- *防火墙自身免于渗透。*防火墙自身是一种与网络连接的设备，如果设计或安装不当，将可能危及安全，在这种情况下它仅提供了一种安全的假象（这比根本没有防火墙更糟糕！）。

Cisco 和 Check Point 是当今两个领先的防火墙厂商。你也能够容易地从 Linux 套件使

用 iptables（通常与 Linux 装在一起的公共域软件）产生一个防火墙（分组过滤器）。此外，如第 4 章和第 5 章中所讨论的，防火墙现在经常在路由器中实现并使用 SDN 进行远程控制。

防火墙能够分为 3 类：**传统分组过滤器**（traditional packet filter）、**状态过滤器**（stateful filter）和**应用程序网关**（application gateway）。在下面小节中，我们将依次学习它们。

1. 传统的分组过滤器

如图 8-33 所示，一个机构通常都有一个将其内部网络与其 ISP（并因此与更大的公共因特网相连）相连的网关路由器。所有离开和进入内部网络的流量都要经过这个路由器，而这个路由器正是**分组过滤**（packet filtering）出现的地方。分组过滤器独立地检查每个数据报，然后基于管理员特定的规则决定该数据报应当允许通过还是应当丢弃。过滤决定通常基于下列因素：

- IP 源或目的地址。
- 在 IP 数据报中的协议类型字段：TCP、UDP、ICMP、OSPF 等。
- TCP 或 UDP 的源和目的端口。
- TCP 标志比特：SYN、ACK 等。
- ICMP 报文类型。
- 数据报离开和进入网络的不同规则。
- 对不同路由器接口的不同规则。

网络管理员基于机构的策略配置防火墙。该策略可以考虑用户生产率和带宽使用以及对一个机构的安全性关注。表 8-5 列出了一个机构可能具有的若干可能的策略，以及它们是如何用一个分组过滤器来处理分组的。例如，如果该机构除了允许访问它的公共 Web 服务器外不希望任何入 TCP 连接的话，那么它能够阻挡所有的入 TCP SYN 报文段，但具有目的地端口 80 的 TCP SYN 报文段除外，并且该目的 IP 地址对应于该 Web 服务器。如果该机构不希望它的用户用因特网无线电应用独占访问带宽，那么它能够阻挡所有非关键性 UDP 流量（因为因特网无线电经常是通过 UDP 发送的）。如果该机构不希望它的内部网络被外部绘制结构图（被跟踪路由），那么它能够阻挡所有 ICMP TTL 过期的报文离开该机构的网络。

表 8-5　对于 Web 服务器在 130.207.244.203 的某机构网络 130.207/16，其策略和对应的过滤规则

策略	防火墙设置
无外部 Web 访问	丢弃所有到任何 IP 地址、端口 80 的出分组
无入 TCP 连接，但那些只访问机构公共 Web 服务器的分组除外	丢弃所有到除 130.207.244.203、端口 80 外的任何 IP 地址的入 TCP SYN 分组
防止 Web 无线电占据可用带宽	丢弃所有入 UDP 分组，但 DNS 分组除外
防止你的网络被用于一个 smurf DoS 攻击	丢弃所有去往某“广播”地址（例如 130.207.255.255）的 ICMP ping 分组
防止你的网络被跟踪路由	丢弃所有出 ICMP TTL 过期流量

一条过滤策略能够基于地址和端口号的结合。例如，一台过滤路由器能够转发所有 Telnet 数据报（那些具有端口号 23 的数据报），但那些包括在一个特定的 IP 地址列表中的去往和来自的地址除外。这些策略允许在许可列表上的地址进行 Telnet 连接。不幸的是，

基于外部地址的策略无法对其源地址被哄骗的数据报提供保护。

过滤也可根据TCP ACK比特是否设置来进行。如果一个机构要使内部客户连接到外部服务器，却要防止外部客户连接到内部服务器，这个技巧很有效。3.5节讲过，在每个TCP连接中第一个报文段的ACK比特都设为0，而连接中的所有其他报文段的ACK比特都设为1。因此，如果一个机构要阻止外部客户发起到内部服务器的连接，就只需直接过滤进入的所有ACK比特设为0的报文段。这个策略去除了所有从外部发起的所有TCP连接，但是允许内部发起TCP连接。

在路由器中使用访问控制列表实现防火墙规则，每个路由器接口有它自己的列表。表8-6中显示了对于某机构222.22/16的访问控制列表的例子。该访问控制列表适用于将路由器与机构外部ISP连接的某个接口。这些规则被应用到通过该接口自上而下传递的每个数据报。前两条规则一起允许内部用户在Web上冲浪：第一条规则允许任何具有目的端口80的TCP分组离开该机构网络；第二条规则允许任何具有源端口80且ACK比特置位的TCP分组进入该机构网络。注意到如果一个外部源试图与一台内部主机建立一条TCP连接，该连接将被阻挡，即使该源或目的端口为80。接下来的两条规则一起允许DNS分组进入和离开该机构网络。总而言之，这种限制性相当强的访问控制列表阻挡所有流量，但由该机构内发起的Web流量和DNS流量除外。[CERT Filtering 2012]提供了一个推荐的端口/协议分组过滤的列表，以避免在现有网络应用中的一些周知的安全性漏洞。

表8-6 用于某路由器接口的访问控制列表

动作	源地址	目的地址	协议	源端口	目的端口	标志比特
允许	222.22/16	222.22/16的外部	TCP	>1023	80	任意
允许	222.22/16的外部	222.22/16	TCP	80	>1023	ACK
允许	222.22/16	222.22/16的外部	UDP	>1023	53	—
允许	222.22/16的外部	222.22/16	UDP	53	>1023	—
拒绝	全部	全部	全部	全部	全部	全部

2. 状态分组过滤器

在传统的分组过滤器中，根据每个分组分离地做出过滤决定。状态过滤器实际地跟踪TCP连接，并使用这种知识作出过滤决定。

为了理解状态过滤器，我们来重新审视表8-6中的访问控制列表。尽管限制性相当强，表8-6中的访问控制列表仍然允许来自外部的ACK=1且源端口为80的任何分组到达，通过该过滤器。这样的分组能够被试图用异常分组来崩溃内部系统、执行拒绝服务攻击或绘制内部网络的攻击者使用。幼稚的解决方案是也阻挡TCP ACK分组，但是这样的方法将妨碍机构内部的用户在Web上冲浪。

状态过滤器通过用一张连接表来跟踪所有进行中的TCP连接来解决这个问题。这种方法是可能的：因为防火墙能够通过观察三次握手（SYN、SYNACK和ACK）来观察一条新连接的开始；而且当它看到该连接的一个FIN分组时，它能够观察该连接的结束。当防火墙经过比如说60秒还没有看到该连接的任何活动性，它也能够（保守地）假设该连接结束了。某防火墙的一张连接表例子显示在表8-7中。这张连接表指示了当前有3条进行中的TCP连接，所有的连接都是从该机构内部发起的。此外，该状态过滤器在它的访问控制列表中包括了一个新栏，即“核对连接”，如表8-8中所示。注意到表8-8

与表8-6中的访问控制列表相同，只是此时它指示应当核对其中两条规则所对应的连接。

表8-7 用于状态过滤器的连接表

源地址	目的地址	源端口	目的端口
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

表8-8 用于状态过滤器的访问控制列表

动作	源地址	目的地址	协议	源端口	目的端口	标志比特	核对连接
允许	222.22/16	222.22/16的外部	TCP	>1023	80	任意	
允许	222.22/16的外部	222.22/16	TCP	80	>1023	ACK	X
允许	222.22/16	222.22/16的外部	UDP	>1023	53	—	
允许	222.22/16的外部	222.22/16	UDP	53	>1023	—	X
拒绝	全部	全部	全部	全部	全部	全部	

我们浏览某些例子来看看连接表和扩展的访问控制列表是如何联手工作的。假设一个攻击者通过发送一个具有TCP源端口80和ACK标志置位的数据报，试图向机构网络中发送一个异常分组。进一步假设该分组具有源端口号12543和源IP地址150.23.23.155。当这个分组到防火墙时，防火墙核对表8-8中的访问控制列表，该表指出在允许该分组进入机构网络之前还必须核对连接表。该防火墙正确地核对了连接表，发现这个分组不是某进行中的TCP连接的一部分，从而拒绝了该分组。举第二个例子，假设一个内部的用户要在外部Web站点冲浪。因为该用户首先发送了一个TCP SYN报文段，所以该用户的TCP连接在连接表中有了记录。当Web服务器发送回分组（ACK比特进行了必要的设置），该防火墙核对了连接表并明白一条对应的连接在进行中。防火墙因此将让这些分组通过，从而不会干扰内部用户的Web冲浪活动。

3. 应用程序网关

在上面的例子中，我们已经看到了分组级过滤使得一个机构可以根据IP的内容和TCP/UDP首部（包括IP地址、端口号和ACK比特）执行粗粒度过滤。但是如果一个机构仅为一个内部用户的受限集合（与IP地址情况正相反）提供Telnet服务该怎样做呢？如果该机构要这些特权用户在允许创建向外部的Telnet会话之前首先鉴别他们自己该怎样做呢？这些任务都超出了传统过滤器和状态过滤器的能力。的确，有关内部用户的身份信息是应用层数据，并不包括在IP/TCP/UDP首部中。

为了得到更高水平的安全性，防火墙必须把分组过滤器和应用程序网关结合起来。应用程序网关除了看IP/TCP/UDP首部外，还基于应用数据来做策略决定。一个**应用程序网关**（application gateway）是一个应用程序特定的服务器，所有应用程序数据（入和出的）都必须通过它。多个应用程序网关可以在同一主机上运行，但是每一个网关都是有自己的进程的单独服务器。

为了更深入地了解应用程序网关，我们来设计一个防火墙，它只允许内部客户的受限集合向外Telnet，不允许任何外部客户向内Telnet。这一策略可通过将分组过滤（在一台

路由器上）和一个 Telnet 应用程序网关结合起来实现，如图 8-34 所示。路由器的过滤器配置为阻塞所有 Telnet 连接，但从该应用程序网关 IP 地址发起的连接除外。这样的过滤器配置迫使所有向外的 Telnet连接都通过应用程序网关。现在考虑一个要向外部 Telnet 的内部用户。这个用户必须首先和应用程序网关建立一个 Telnet 会话。在网关（网关监听进入的 Telnet 会话）上一直运行的应用程序提示用户输入用户 ID 和口令。当这个用户提供这些信息时，应用程序网关检查这个用户是否得到许可向外 Telnet。如果没有，网关则中止这个内部用户向该网关发起的 Telnet 连接。如果该用户得到许可，则这个网关：①提示用户输入它所要连接的外部主机的主机名；②在这个网关和某外部主机之间建立一个 Telnet 会话；③将从这个用户到达的所有数据中继到该外部主机，并且把来自这个外部主机的所有数据都中继给这个用户。所以，该 Telnet 应用程序网关不仅执行用户授权，而且同时充当一个 Telnet 服务器和一个 Telnet 客户，在这个用户和该远程 Telnet 服务器之间中继信息。注意到过滤器因为该网关发起向外部的 Telnet 连接，将允许执行步骤②。

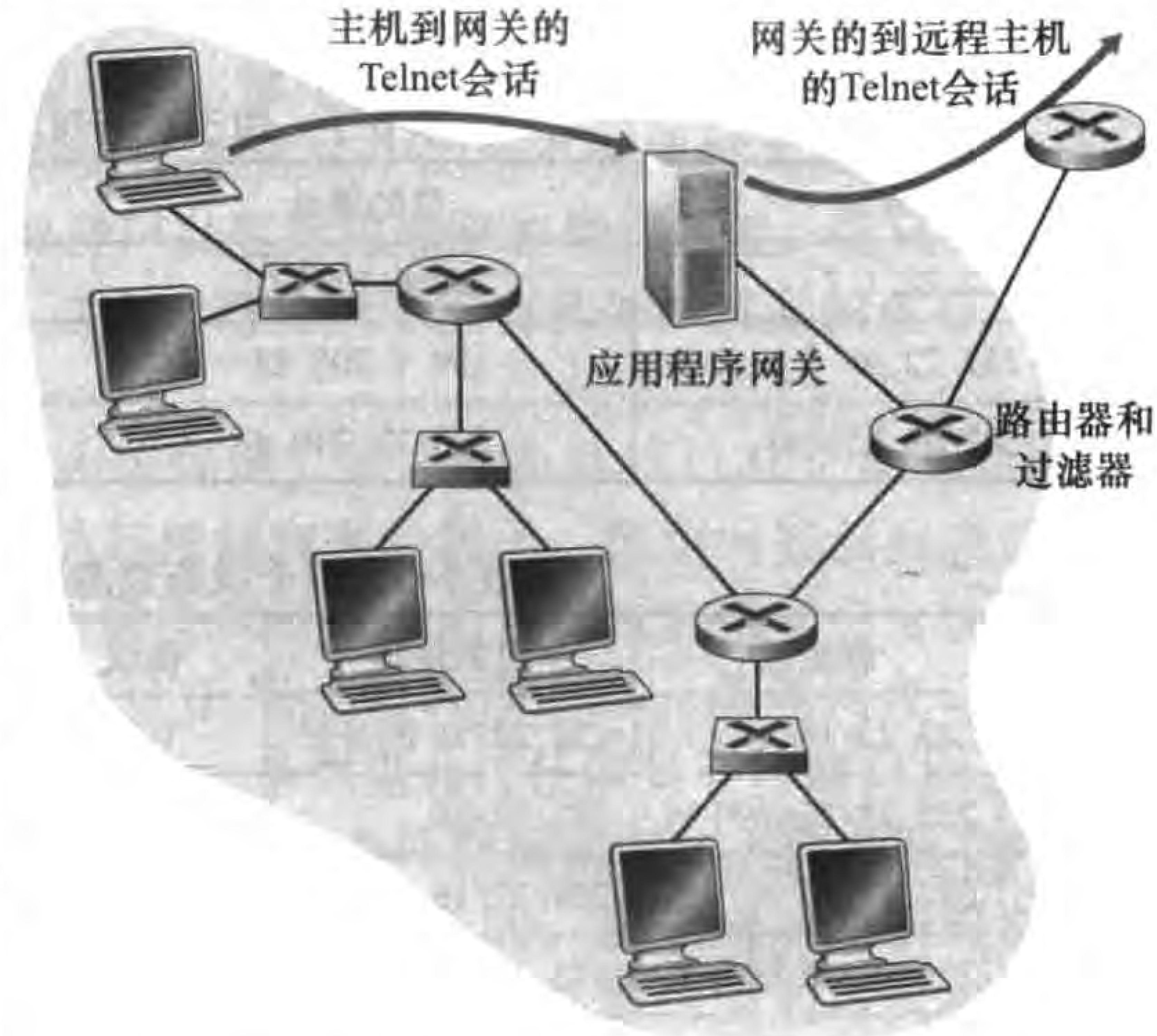


图 8-34　由应用程序网关和过滤器组成的防火墙

历史事件

匿名与隐私

假定你要访问一个有争议的 Web 网站（例如某政治活动家的网站），并且你：①不想向该 Web 网站透漏你的 IP 地址；②不想要你的本地 ISP（它可能是你住家或办公室的 ISP）知道你正在访问该站点；③不想要你的本地 ISP 看到你正在与该站点交换的数据。如果你使用传统的方法直接与该 Web 站点连接而没有任何加密，你无法实现这三个诉求。即使你使用 SSL，你也无法实现前两个诉求：你的源 IP 地址呈现在你发送给 Web 网站的每个数据报中；你发送的每个分组的目的地址能够容易地被你本地 ISP 嗅探到。

为了获得隐私和匿名，你能够使用如图 8-35 所示的一种可信代理服务器和 SSL 的组合。利用这种方法，你首先与可信代理建立一条 SSL 连接。然后你在该 SSL 连接中向所希望站点的网页发送一个 HTTP 请求。当代理接收到该 SSL 加密的 HTTP 请求，它解密请求并向 Web 站点转发该明文 HTTP 请求。接下来 Web 站点响应该代理，该代理经过 SSL 再向你转发该响应。因为该 Web 站点仅看到代理的 IP 地址，并非你的客户 IP 地址，你的确获得了对该 Web 站点的匿名访问。并且因为你和代理之间的所有流量均被加密，你的本地 ISP 无法通过对你访问的站点做日志和记录你交换的数据来侵犯你的隐私。今天许多公司（例如 proxify. com）提供了这种代理服务。

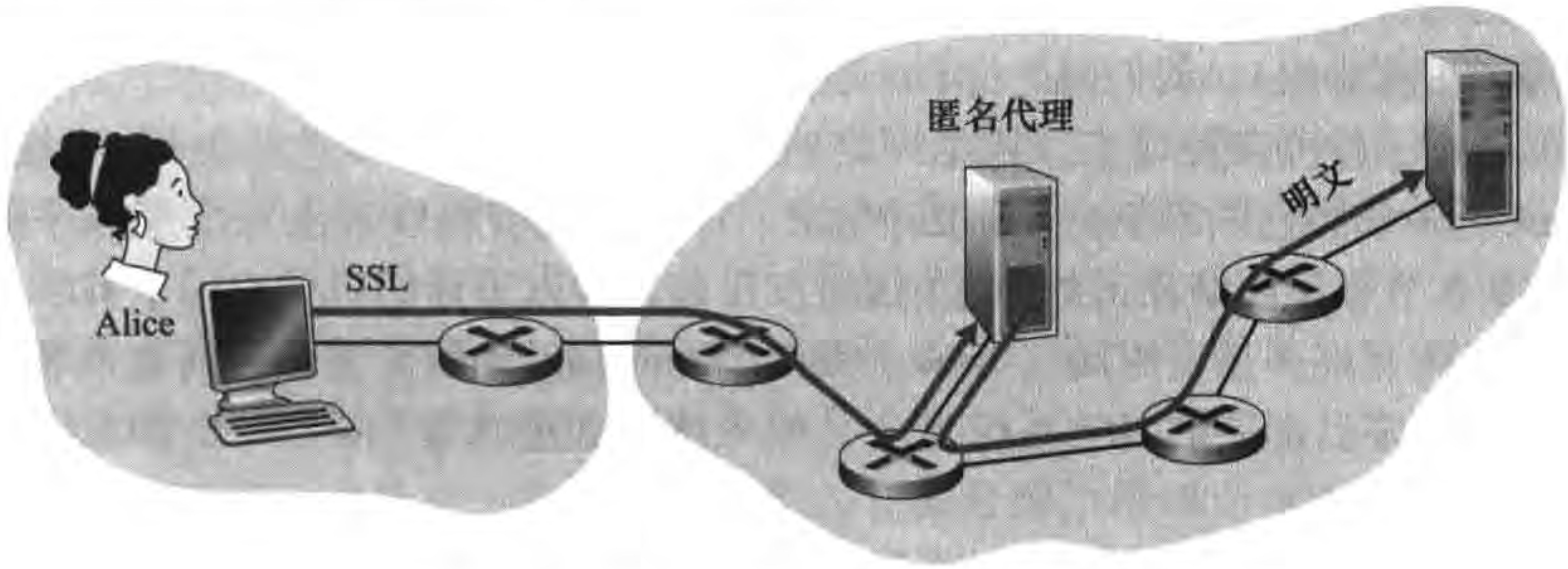


图 8-35 利用代理提供匿名和隐私

当然，在这个解决方案中，你的代理知道一切：它知道你的 IP 地址和你正在冲浪的站点的 IP 地址；并且它能够看到你与该 Web 站点之间以明文形式交换的所有流量。因此，这种解决方案的好坏取决于该代理的可信度。由 TOR 匿名和隐私服务所采用的一种更为健壮的方法是，让你的流量路由通过一系列“不串通”的代理服务器［TOR 2012］。特别是，TOR 允许独立的个体向其代理池贡献代理。当某用户使用 TOR 与一个服务器连接，TOR 随机地（从它的代理池）选择一条三个代理构成的链，并通过该链在客户和服务器之间路由所有流量。以这种方式，假设这些代理并不串通，无人知道在你的 IP 地址和目标 Web 站点之间发生的通信。此外，尽管在最后的代理和服务器之间发送明文，但这个最后代理并不知道哪个 IP 地址正在发送和接收明文。

内部网络通常有多个应用程序网关，例如 Telnet、HTTP、FTP 和电子邮件网关。事实上，一个机构的邮件服务器（见 2.3 节）和 Web 高速缓存都是应用程序网关。

应用程序网关也有其缺陷。首先，每一个应用程序都需要一个不同的应用程序网关。第二，因为所有数据都由网关转发，付出的性能负担较重。当多个用户或应用程序使用同一个网关计算机时，这成为特别值得关注的问题。最后，当用户发起一个请求时，客户软件必须知道如何联系这个网关，并且必须告诉应用程序网关如何连接到哪个外部服务器。

8.9.2 入侵检测系统

我们刚刚看到，当决定让哪个分组通过防火墙时，分组过滤器（传统的和状态的）检查 IP、TCP、UDP 和 ICMP 首部字段。然而，为了检测多种攻击类型，我们需要执行**深度分组检查**（deep packet inspection），即查看首部字段以外部分，深入查看分组携带的实际应用数据。如我们在 8.9.1 节所见，应用程序网关经常做深度分组检查。而一个应用程序网关仅对一种特定的应用程序执行这种检查。

显然，这为另一种设备提供了商机，即一种不仅能够检查所有通过它传递的分组的首部（类似于分组过滤器），而且能执行深度分组检查（与分组过滤器不同）的设备。当这样的设备观察到一个可疑的分组时，或一系列可疑的分组时，它能够防止这些分组进入该机构网络。或者仅仅是因为觉得该活动可疑，该设备虽说能够让这些分组通过，但要向网络管理员发出告警，网络管理员然后密切关注该流量并采取适当的行动。当观察到潜在恶意流量时能产生告警的设备称为**入侵检测系统**（Intrusion Detection System，IDS）。滤除可疑流量的设备称为**入侵防止系统**（Intrusion Prevention System，IPS）。在本节中我们一起学习 IDS 和 IPS 这

两种系统，因为这些系统的最为有趣的技术方面是它们检测可疑流量的原理（而不是它们是否发送告警或丢弃分组）。我们因此将 IDS 系统和 IPS 系统统称为 IDS 系统。

IDS 能够用于检测多种攻击，包括网络映射（例如使用 nmap 进行分析）、端口扫描、TCP 栈扫描、DoS 带宽洪泛攻击、蠕虫和病毒、操作系统脆弱性攻击和应用程序脆弱性攻击。（参见 1.6 节有关网络攻击的概述内容。）目前，数以千计的机构应用了 IDS 系统。这些已部署的系统有许多是专用的，Cisco、Check Point 和其他安全装备厂商在市场上销售这些系统。但是许多已部署的 IDS 系统是公共域系统，如极为流行的 Snort IDS 系统（我们将简要讨论它）。

一个机构可能在它的机构网络中部署一个或多个 IDS 传感器。图 8-36 显示了一个具有 3 个 IDS 传感器的机构。当部署了多个传感器时，它们通常共同工作，向一个中心 IDS 处理器发送有关可疑流量活动的信息，中心处理器收集并综合这些信息，当认为适合时向网络管理员发送告警。在图 8-36 中，该机构将其网络划分为两个区域：一个高度安全区域，由分组过滤器和应用程序网关保护，并且由 IDS 系统监视；一个较低度安全区域（称之为**非军事区**（DeMilitarized Zone，DMZ）），该区域仅由分组过滤器保护，但也由 IDS 系统监视。注意到 DMZ 包括了该机构需要与外部通信的服务器，如它的公共 Web 服务器和它的权威 DNS 服务器。

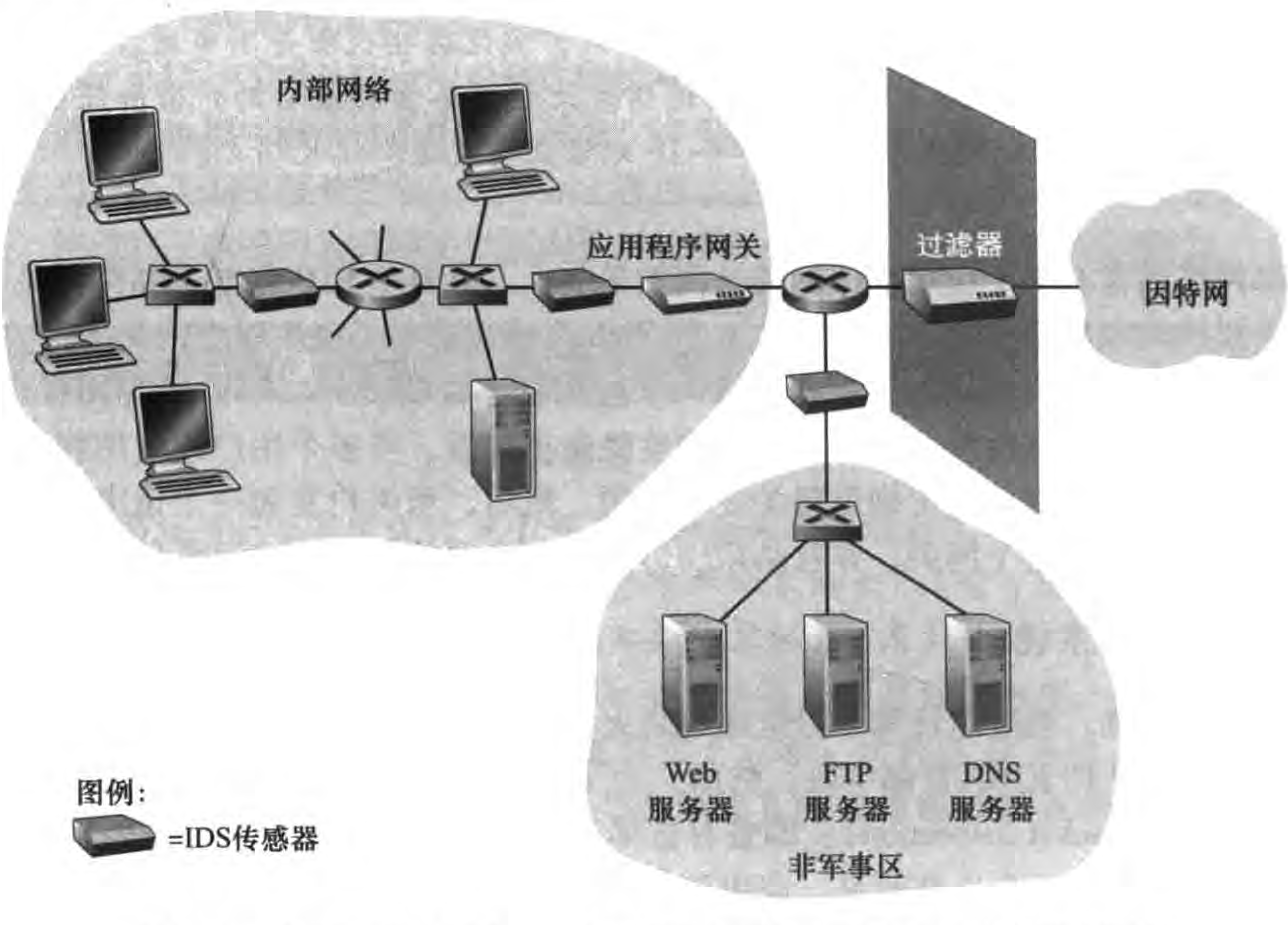


图 8-36 部署一个过滤器、一个应用程序网关和多个 IDS 传感器的机构

此时你也许想知道，为什么使用多个 IDS 传感器？为什么在图 8-36 中不只是在分组过滤器后面放置一个 IDS 传感器（或者甚至与分组过滤器综合）？我们将很快看到，IDS 不仅需要做深度分组检查，而且必须要将每个过往的分组与数以万计的“特征（signature）”进行比较；这可能导致极大的处理量，特别是如果机构从因特网接收每秒数十亿比特的流量时更是如此。将 IDS 传感器进一步向下游放置，每个传感器仅看到该机构流量的一部分，维护能够更容易。无论如何，目前有许多高性能 IDS 和 IPS 系统可供使用，许

多机构实际上能够在靠近其接入路由器附近只使用一个传感器。

IDS 系统大致可分类为**基于特征的系统**（signature-based system）或**基于异常的系统**（anomaly-based system）。基于特征的 IDS 维护了一个范围广泛的攻击特征数据库。每个特征是与一个入侵活动相关联的规则集。一个特征可能只是有关单个分组的特性列表（例如源和目的端口号、协议类型和在分组载荷中的特定比特串），或者可能与一系列分组有关。这些特征通常由研究了已知攻击、技艺熟练的网络安全工程师生成。一个机构的网络管理员能够定制这些特征或者将其加进数据库中。

从运行上讲，基于特征的 IDS 嗅探每个通过它的分组，将每个嗅探的分组与数据库中的特征进行比较。如果某分组（或分组序列）与数据库中的一个特征相匹配，IDS 产生一个告警。该告警能够发送一个电子邮件报文给网络管理员，能够发送给网络管理系统，或只是做日志以供以后检查。

尽管基于特征的 IDS 系统部署广泛，但仍具有一些限制。更重要的是，它们要求根据以前的攻击知识来产生一个准确的特征。换言之，基于特征的 IDS 对不得不记录的新攻击完全缺乏判断力。另一个缺点是，即使与一个特征匹配，它也可能不是一个攻击的结果，因此产生了一个虚假告警。最后，因为每个分组必须与范围广泛的特征集合相比较，IDS 可能处于处理过载状态并因此难以检测出许多恶意分组。

当基于异常的 IDS 观察正常运行的流量时，它会生成一个流量概况文件。然后，它寻找统计上不寻常的分组流，例如，ICMP 分组不寻常的百分比，或端口扫描和 ping 掠过导致指数性突然增长。基于异常的 IDS 系统最大的特点是它们不依赖现有攻击的以前知识。在另一方面，区分正常流量和统计异常流量是一个极具挑战性的问题。迄今为止，大多数部署的 IDS 主要是基于特征的，尽管某些 IDS 包括了某些基于异常的特性。

Snort

Snort 是一种公共域开放源码的 IDS，现有部署达几十万［Snort 2012；Koziol 2003］。它能够运行在 Linux、UNIX 和 Windows 平台上。它使用了通用的嗅探接口 libpcap，Wireshark 和许多其他分组嗅探器也使用了 libpcap。它能够轻松地处理 100Mbps 的流量；安装在千兆比特/秒流量速率下工作，需要多个 Snort 传感器。

为了对 Snort 有一些认识，我们来看一个 Snort 特征的例子：

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ICMP PING NMAP"; dsize: 0; itype: 8;)
```

这个特征由从外部（$EXTERNAL_NET）进入机构网络（$HOME_NET）的任何 ICMP 分组所匹配，其类型是 8（ICMP ping）并且具有空负载（dsize = 0）。因为 nmap（参见 1.6 节）用这些特定的特征产生这些 ping 分组，所以设计出该特征来检测 nmap 的 ping 扫描。当某分组匹配该特征时，Snort 产生一个包括“ICMP PING NAMP”报文的告警。

也许关于 Snort 印象最为深刻的是巨大的用户社区和维护其特征数据库的安全专家。通常在一个新攻击出现后的几个小时内，Snort 社区就编写并发布一个攻击特征，然后它就能被分布在全世界的数十万 Snort 部署者下载。此外，使用 Snort 特征的语法，网络管理员能够根据他们自己的机构需求，通过修改现有的特征或通过创建全新的特征来裁剪某个特征。

8.10 小结

在本章中，我们考察了秘密情人 Bob 和 Alice 能够用于安全通信的各种机制。我们看

到 Bob 和 Alice 对下列因素感兴趣：机密性（因此只有他们才能理解传输的报文内容）、端点鉴别（因此他们确信正在与对方交谈）和报文完整性（因此他们确信在传输过程中他们的报文未被篡改）。当然，安全通信的需求并不限于秘密情人。的确，我们在 8.5 ~ 8.8 节中看到，可以在网络体系结构中的各个层次使用安全性，使之免受采用各种各样攻击手段的坏家伙们的侵扰。

本章前面部分给出了安全通信所依赖的各种原理。在 8.2 节中，我们涉及了加密和解密数据的密码技术，包括对称密钥密码和公开密钥密码。作为今天网络中两种重要的密码技术的特定的学习案例，我们考察了 DES 和 RSA。

在 8.3 节中，我们研究了提供报文完整性的两种方法：报文鉴别码（MAC）和数字签名。这两种方法有一些共同之处。它们都使用了密码散列函数，这两种技术都使我们能够验证报文的源以及报文自身的完整性。一个重要的差异是 MAC 不依赖于加密，而数字签名要求公钥基础设施。如我们在 8.5 ~ 8.8 节所见，这两种技术广泛在实际中都得到了广泛应用。此外，数字签名用于生成数字证书，数字证书对于证实公钥的合法性是重要的。在 8.4 节中，我们考察了端点鉴别并引入了不重数以防御重放攻击。

在 8.5 ~ 8.8 节中，我们研究了几种在实践中得到广泛使用的安全性网络协议。我们看到了对称密钥密码在 PGP、SSL、IPsec 和无线安全性中的核心地位。我们看到了公开密钥密码对 PGP 和 SSL 是至关重要的。我们看到 PGP 使用数字签名而 SSL 和 IPsec 使用 MAC 来保证报文完整性。在目前理解了密码学的基本原理以及学习了这些原理的实际应用方法之后，你现在已经有能力设计你自己的安全网络协议了！

利用 8.2 ~ 8.4 节所包含的技术，Bob 和 Alice 就能够安全通信了。（只希望他们是学习了这些材料的网络专业学生，因此能够使他们的约会不会被 Trudy 发现！）而机密性仅是整个网络安全的一小部分。如我们在 8.9 节中所学习，现在网络安全的焦点越来越多地关注网络基础设施的安全性，以防止“坏家伙”的潜在猛烈攻击。在本章的后面部分，我们因此学习了防火墙和 IDS 系统，它们检查进入和离开一个机构网络的分组。

本章已经涉及了许多基础性问题，同时关注了现代网络安全性中最为重要的主题。希望深入钻研的读者最好研究本章中引用的文献。特别是，我们推荐以下读物：关于攻击和运行安全性的 [Skoudis 2006]，关于密码学及其如何应用于网络安全的 [Kaufman 1995]，有深度且可读性强的关于 SSL 处理的 [Rescorla 2001]，以及透彻地讨论 802.11 安全性且包括对 WEP 及其缺陷的深入研究的 [Edney 2003]。

课后习题和问题

复习题

8.1 节

R1. 报文机密性和报文完整性之间的区别是什么？你能具有机密性而没有完整性吗？你能具有完整性而没有机密性吗？证实你的答案。

R2. 因特网实体（路由器、交换机、DNS 服务器、Web 服务器、用户端系统等）经常需要安全通信。给出三个特定的因特网实体对的例子，它们要安全通信。

8.2 节

R3. 从服务的角度，对称密钥系统和公开密钥系统之间一个重要的差异是什么？

R4. 假定某入侵者拥有一个加密报文以及该报文的解密版本。这个入侵者能够发起已知密文攻击、已知

明文攻击和选择明文攻击吗？

R5. 考虑一个 8 块密码。这个密码有多少种可能的输入块？有多少种可能的映射？如果我们将每种映射视为一个密钥，则该密码具有多少种可能的密钥？

R6. 假定 N 个人中的每个人都和其他 $N-1$ 个人使用对称密钥密码通信。任两人（i 和 j）之间的所有通信对该 N 个人的组中的所有其他人都是可见的，且该组中的其他人都不应当能够解密他们的通信。则这个系统总共需要多少个密钥？现在假定使用公开密钥密码。此时需要多少个密钥？

R7. 假定 $n=10\,000$、$a=10\,023$ 和 $b=10\,004$。请你使用等同的模算术来心算（$a \cdot b$）mod n。

R8. 假设你要通过加密对应于报文 1010111 的十进制数来加密该报文。该十进制数是什么？

8.3~8.4 节

R9. 散列以何种方式提供比检验和（如因特网检验和）更好的报文完整性检验？

R10. 你能够“解密”某报文的散列来得到初始报文吗？解释你的答案。

R11. 考虑 MAC 算法（图 8-9）的一种变形算法，其中发送方发送（m，$H(m)+s$），这里 $H(m)+s$ 是 $H(m)$ 和 s 的级联。该变形算法有缺陷吗？为什么？

R12. 一个签名的文档是可鉴别的和不可伪造的，其含义是什么？

R13. 公钥加密的报文散列以何种方式比使用公钥加密报文提供更好的数字签名？

R14. 假设 certifier. com 生成一个用于 foo. com 的证书。通常整个证书将用 certifier. com 的公钥加密。这种说法是正确还是错误？

R15. 假设 Alice 有一个准备发送给任何请求者的报文。数以千计的人要获得 Alice 的报文，但每个人都要确保该报文的完整性。在这种场景下，你认为是基于 MAC 还是基于数字签名的完整性方案更为适合？为什么？

R16. 在某端点鉴别协议中，使用不重数的目的是什么？

R17. 我们说一个不重数是一个在生存期中只使用一次的值，这意味着什么？其中是指谁的生存期？

R18. 基于 HMAC 的报文完整性方案易受重放攻击影响吗？如果是，能够在方案中综合一个不重数来去除这种脆弱性吗？

8.5~8.8 节

R19. 假定 Bob 从 Alice 处接收一个 PGP 报文。Bob 怎样才能确定 Alice（而不是如 Trudy）生成了该报文？PGP 为保证报文完整性使用了 MAC 吗？

R20. 在 SSL 记录中，有一个字段用于 SSL 序号。这种说法是正确还是错误？

R21. 在 SSL 握手中随机不重数的目的是什么？

R22. 假设某 SSL 会话应用了具有 CBC 的块密码。服务器以明文向客户发送了 IV。这种说法是正确还是错误？

R23. 假设 Bob 向 Trudy 发起一条 TCP 连接，而 Trudy 正在伪装她是 Alice。在握手期间，Trudy 向 Bob 发送 Alice 的证书。在 SSL 握手算法的哪一步，Bob 将发现他没有与 Alice 通信？

R24. 考虑使用 IPsec 从主机 A 向主机 B 发送分组流。通常，为该流中的每个发送分组将创建一个新 SA。这种说法是正确还是错误？

R25. 假设在图 8-28 中总部和分支机构之间通过 IPsec 运行 TCP。如果 TCP 重新传输相同的分组，则由 R1 发送的两个对应的分组将在 ESP 首部中具有相同的序号。这种说法是正确还是错误？

R26. IKE SA 和 IPsec SA 是相同的东西。这种说法是正确还是错误？

R27. 考虑 802.11 的 WEP。假定数据是 10101100 并且密钥流是 11110000。相应的密文是什么？

R28. 在 WEP 中，在每个帧中以明文发送 IV。这种说法是正确还是错误？

8.9 节

R29. 状态分组过滤器维护两个数据结构。给出它们的名字并简单地讨论它们做些什么。

R30. 考虑某传统的（无状态的）分组过滤器。该分组过滤器可能基于 TCP 标志位以及其他首部字段过滤分组。这种说法是正确还是错误？

R31. 在传统的分组过滤器中，每个接口能够具有自己的访问控制表。这种说法是正确还是错误？

R32. 为什么应用程序网关必须与分组过滤器协同工作才能有效？

R33. 基于特征的IDS和IPS检查TCP和UDP报文段的载荷。这种说法是正确还是错误?

习题

P1. 使用图8-3中的单码代替密码，加密报文“This is an easy problem”，并解密报文“rmij'u uamu xyj”。

P2. Trudy使用了已知明文攻击，其中她知道了7个字母的（密文，明文）转换对，减少了8.2.1节的例子中将被检查的大约10^9数量级的可能替换的数量。请说明之。

P3. 考虑图8-4所示的多码代替密码系统。利用报文“The quick brown fox jumps over the lazy dogs”得到的明文编码，选择明文攻击足以破解所有报文吗？为什么？

P4. 考虑图8-5中显示的块密码。假设每个块密码T_i只是反转了8个输入比特的次序（例如，使得11110000变为00001111）。进一步假设64比特置乱函数不修改任何比特（使得第m个比特的输出值等于第m个比特的输入值）。

a. 对于$n=3$和初始64比特输入等于10100000重复了8次，输出的值是多少?

b. 重复（a），但此时将初始64比特的最后一个比特从0变为1。

c. 重复（a）和（b），但此时假定64比特的置乱函数反转了64比特的次序。

P5. 考虑图8-5中的块密码。对于给定的“密钥”，Alice和Bob将需要8个表，每张表8比特乘以8比特。对于Alice（或Bob）来说，要存储所有8张表，将需要多少比特的存储器？这个数如何与一个全表64比特的块密码所需的比特数进行比较?

P6. 考虑在表8-1中的3比特块密码。假定明文是100100100。

a. 初始假设未使用CBC。生成的密文是什么?

b. 假设Trudy嗅探该密文。假设她知道正在应用无CBC的一个3比特块密码（但不知道特定的密码），她能够推测到什么?

c. 现在假设使用CBC，其中IV=111。产生的密文是什么?

P7. 如题：

a. 使用RSA，选择$p=3$和$q=11$，采用对每个字母独立地加密的方法加密短语“dog”。对已加密报文应用解密算法恢复出原报文。

b. 重复（a），而此时加密“dog”作为一个报文m。

P8. 考虑具有$p=5$和$q=11$的RSA。

a. n和z是什么?

b. 令e为3。为什么这是一个对e的可接受的选择?

c. 求d使得$de=1 \pmod z$和$d<160$。

d. 使用密钥(n, e)加密报文$m=8$。令c表示对应的密文。显示所有工作。提示：为了简化计算，使用如下事实。

$$[(a \bmod n)\cdot(b \bmod n)] \bmod n = (a\cdot b) \bmod n$$

P9. 在这个习题中，我们探讨Diffie-Hellman(DH)公钥加密算法，该算法允许两个实体协商一个共享的密钥。该DH算法利用一个大素数p和另一个小于p的大数g。p和g都是公开的（因此攻击者将知道它们）。在DH中，Alice和Bob每人分别独立地选择秘密密钥S_A和S_B。Alice则通过将g提高到S_A并以p为模来计算她的公钥T_A。类似地，Bob则通过将g提高到S_B并以p为模来计算他的公钥T_B。此后Alice和Bob经过因特网交换他们的公钥。Alice则通过将T_B提高到S_A并以p为模来计算出共享密钥S。类似地，Bob则通过将T_A提高到S_B并以p为模来计算出共享密钥S'。

a. 证明在一般情况下，Alice和Bob得到相同的对称密钥，即证明$S=S'$。

b. 对于$p=11$和$g=2$，假定Alice和Bob分别选择私钥$S_A=5$和$S_B=12$，计算Alice和Bob的公钥T_A和T_B。显示所有计算过程。

c. 接着（b），现在计算共享对称密钥S。显示所有计算过程。

d. 提供一个时序图，显示Diffie-Hellman是如何能够受到中间人攻击的。该时序图应当具有3条垂直

线，分别对应 Alice、Bob 和攻击者 Trudy。

P10. 假定 Alice 要与采用对称密钥密码体制的 Bob 使用一个会话密钥 K_S 通信。在 8.2 节中，我们知道了如何使用公开密钥密码从 Alice 向 Bob 分发该会话密钥。在本习题中，我们探讨不使用公开密钥密码而使用一个密钥分发中心（KDC）分发会话密钥的方法。KDC 是一个与每个注册用户共享独特的秘密对称密钥的服务器。对于 Alice 和 Bob 而言，$K_{A\text{-}KDC}$ 和 $K_{B\text{-}KDC}$ 表示了这些密钥。设计一个使用 KDC 向 Alice 和 Bob 分发 K_S 的方案。你的方案应当使用三种报文来分发会话密钥：一种从 Alice 到 KDC 的报文；一种从 KDC 到 Alice 的报文；最后一种是从 Alice 到 Bob 的报文。第一种报文为 $K_{A\text{-}KDC}(A, B)$。使用标记 $K_{A\text{-}KDC}$、$K_{B\text{-}KDC}$、S、A 和 B 回答下列问题。

a. 第二种报文是什么？

b. 第三种报文是什么？

P11. 计算一个不同于图 8-8 中的两个报文的第三个报文，使该报文具有与图 8-8 中的报文相同的检验和。

P12. 假定 Alice 和 Bob 共享两个秘密密钥：一个鉴别密钥 S_1 和一个对称加密密钥 S_2。扩充图 8-9，使之提供完整性和机密性。

P13. 在 BitTorrent P2P 文件分发协议中（参见第 2 章），种子将文件分割为块，并且对等方彼此重分发这些块。不使用任何保护，一个攻击者能够容易地通过假冒善意的对等方并向洪流中的一部分对等方发送假冒块来实施破坏。这些未被怀疑的对等方则重新向其他对等方发送这些假冒块，其他对等方则将再次向甚至更多的对等方重新分发这些假冒块。因此，对于 BitTorrent 来说，采用一种机制使对等方能验证一个块的完整性，从而使得假冒块无法分发，这是至关重要的。假设当某对等方加入一个洪流时，它初始从一个完全受信任的源得到一个 .torrent 文件。描述允许对等方验证块完整性的一个简单的方案。

P14. OSPF 路由选择协议使用一个 MAC 而不是数字签名来提供报文完整性。你认为选择 MAC 而未选择数字签名的原因是什么？

P15. 考虑图 8-18 中的鉴别协议，其中 Alice 向 Bob 鉴别她自己，我们看来工作正常（即我们没有发现其中有缺陷）。现在假定当 Alice 向 Bob 鉴别她自己的同时，Bob 必须向 Alice 鉴别他自己。给出一个情况，此时 Trudy 假装是 Alice，向 Bob 鉴别她自己是Alice。（提示：该协议运行的顺序，鉴别过程可由 Trudy 或 Bob 发起，能够任意地交织在一起。特别注意 Bob 和 Alice 将使用不重数这样一个事实，如果不小心的话，能够恶意地使用相同的不重数。）

P16. 一个自然的问题是我们能否使用一个不重数的公钥密码来解决 8.4 节中的端点鉴别问题。考虑下列自然的协议：①Alice 向 Bob 发送报文 "I am Alice"；②Bob 选择一个不重数并将其发送给 Alice；③Alice使用她的私钥来加密该不重数并向 Bob 发送得到的值；④Bob 对接收到的报文应用 Alice 的公钥。因此，Bob 计算 R 并鉴别了 Alice。

a. 画图表示这个协议，使用本书中应用的公钥和私钥的标记法。

b. 假定未使用证书。描述 Trudy 怎样能够通过拦截 Alice 的报文，进而对 Bob 假装她是 Alice 而成为一名"中间人"。

P17. 图 8-19 显示了 Alice 必须执行 PGP 的操作，以提供机密性、鉴别和完整性。图示出当 Bob 接收来自 Alice 的包时必须执行的对应操作。

P18. 假定 Alice 要向 Bob 发送电子邮件。Bob 具有一个公共－私有密钥对（K_B^+，K_B^-），并且 Alice 具有 Bob 的证书。但 Alice 不具有公钥私钥对。Alice 和 Bob（以及全世界）共享相同的散列函数$H(\cdot)$。

a. 在这种情况下，能设计一种方案使得 Bob 能够验证 Alice 创建的报文吗？如果能，用方框图显示 Alice 和 Bob 是如何做的。

b. 能设计一个对从 Alice 向 Bob 发送的报文提供机密性的方案吗？如果能，用方块图显示 Alice 和 Bob 是如何做的。

P19. 考虑下面对于某 SSL 会话的一部分的 Wireshark 输出。

a. Wireshark 分组 112 是由客户还是由服务器发送的？

b. 服务器的 IP 地址和端口号是什么？

c. 假定没有丢包和重传，由客户发送的下一个 TCP 报文段的序号将是什么？

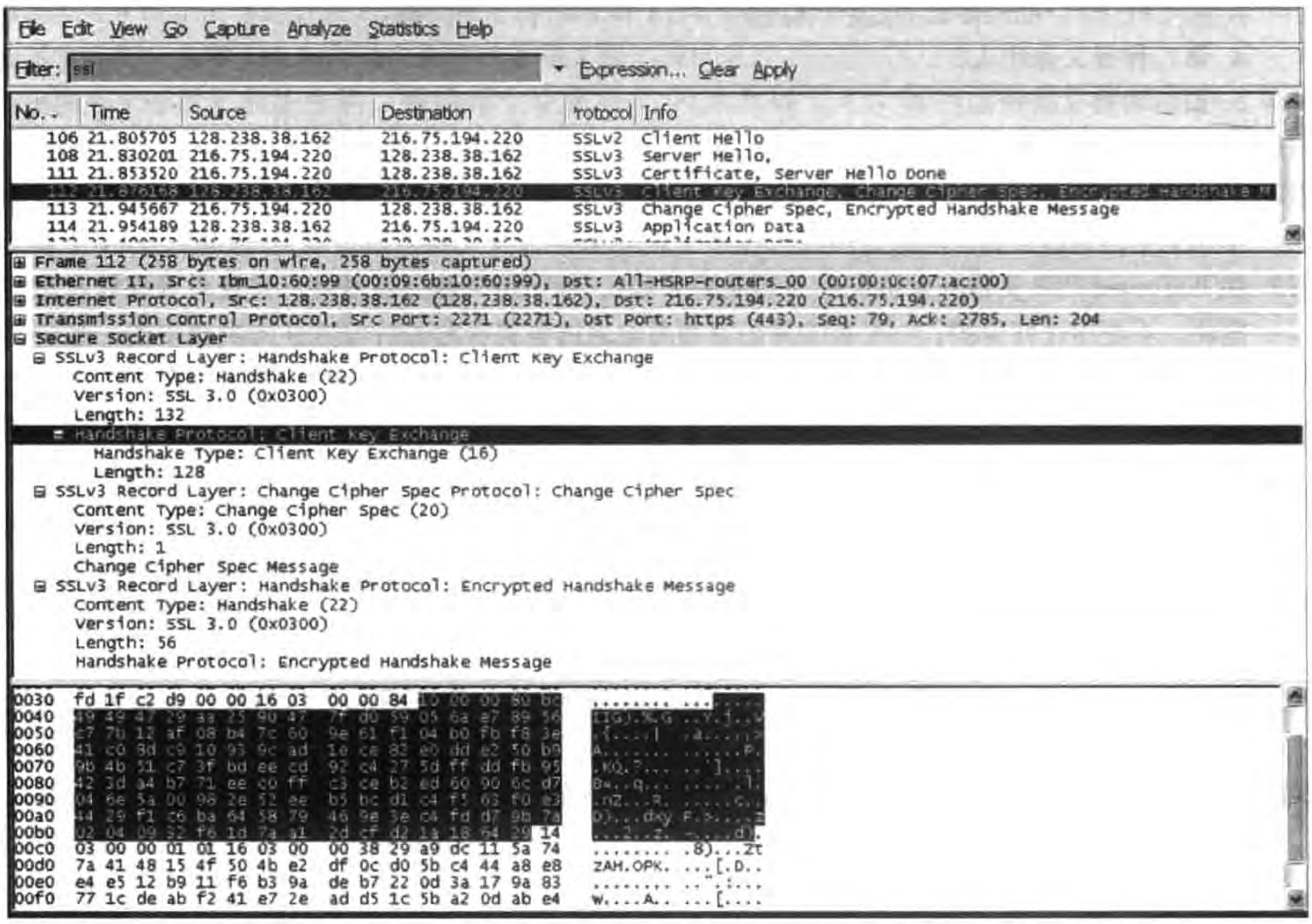
d. Wireshark 分组 112 包含了多少个 SSL 记录？

e. 分组 112 包含了一个主密钥或者一个加密的主密钥吗？或者两者都不是？

f. 假定握手类型字段是 1 字节并且每个长度字段是 3 字节，主密钥（或加密的主密钥）的第一个和最后一个字节的值是什么？

g. 客户加密的握手报文考虑了多少 SSL 记录？

h. 服务器加密的握手报文考虑了多少 SSL 记录？



（Wireshark 屏幕截图的重印获得 Wireshark 基金会的许可）

P20. 8.6.1 节中表明，不使用序号，Trudy（一名中间人）能够在一个 SSL 会话中通过互换 TCP 报文段实施破坏。Trudy 能够通过删除一个 TCP 报文段做某种类似的事情吗？在该删除攻击中，她需要做什么才能成功？它将具有什么影响？

P21. 假定 Alice 和 Bob 通过一个 SSL 会话通信。假定一个没有任何共享密钥的攻击者，在某分组流中插入一个假冒的 TCP 报文段，该报文段具有正确的 TCP 检验和及序号（以及正确的 IP 地址和端口号）。在接收侧 SSL 将接受该假冒分组并传递载荷给接收应用程序吗？为什么？

P22. 下列是有关图 8-28 的判断题。

a. 当在 172.16.1/24 中的主机向一台 Amazon.com 服务器发送一个数据报时，路由器 R1 将使用 IPsec 加密该数据报。

b. 当在 172.16.1/24 中的主机向在 172.16.2/24 中的主机发送一个数据报时，路由器 R1 将改变该 IP 数据报的源和目的地址。

c. 假定在 172.16.1/24 中的主机向在 172.16.2/24 中的 Web 服务器发起一个 TCP 连接。作为此次连接的一部分，由 R1 发送的所有数据报将在 IPv4 首部字段最左边具有协议号 50。

d. 考虑从在 172.16.1/24 中的主机向在 172.16.2/24 中的主机发送一个 TCP 报文段。假定对该报文段的应答丢失了，因此 TCP 重新发送该报文段。因为 IPsec 使用序号，R1 将不重新发送该 TCP 报文段。

P23. 考虑图 8-28 中的例子。假定 Trudy 是中间人，她能够在从 R1 和 R2 发出的数据报流中插入数据报。

作为重放攻击一部分，Trudy 发送一个从 R1 到 R2 发送的数据报的冗余副本。R2 将解密该冗余的数据报并将其转发进分支机构网络吗？如果不是，详细描述 R2 如何检测该冗余的数据报。

P24. 考虑下列伪 WEP 协议。其密钥是 4 比特，IV 是 2 比特。当产生密钥流时，IV 被附加到密钥的后面。假定共享的密钥是 1010。密钥流的 4 个可能输入如下：

101000：0010101101010101001011010100100…

101001：1010011011001010110100100101101…

101010：0001101000111100010100101001111…

101011：1111101010000000101010100010111…

假定所有报文都是 8 比特长。假定 ICV（完整性检查）是 4 比特长，并且通过用数据的后 4 比特异或数据的前 4 比特来计算。假定该伪 WEP 分组由 3 个字段组成：首先是 IV 字段，然后是报文字段，最后是 ICV 字段，这些字段中的某些被加密。

a. 我们希望使用 IV = 11 和 WEP 发送报文 m = 10100000。在这 3 个 WEP 字段中将有什么样的值？
b. 说明当接收方解密该 WEP 分组时，它恢复报文和 ICV。
c. 假定 Trudy 截获了一个 WEP 分组（并不必要使用 IV = 11）并要在向接收方转发前修改该分组。假定 Trudy 翻转了第一个 ICV 比特。假定 Trudy 并不知道用于任何 IV 的密钥流，则 Trudy 也必须翻转哪些其他比特，使得接收到的分组通过 ICV 检查？
d. 通过修改（a）中 WEP 分组中的比特，解密所生成的分组，并验证完整性检查来评价你的答案。

P25. 对于尽可能限制但能实现下列功能的一台有状态防火墙，提供一张过滤器表和一张连接表：

a. 允许所有的内部用户与外部用户创建 Telnet 会话。
b. 允许外部用户冲浪公司位于 222.22.0.12 的 Web 站点。
c. 否则阻挡所有入流量和出流量。

内部网络为 222.22/16。在你的答案中，假设连接表当前缓存了 3 个从内向外的连接。你需要虚构适当的 IP 地址和端口号。

P26. 假设 Alice 要使用 TOR 类似的服务访问 Web 站点 activist.com。该服务使用两个不串通的代理服务器 Proxy1 和 Proxy2。Alice 首先从某个中央服务器获得对 Proxy1 和 Proxy2 的证书（每个都包含一个公钥）。用 $K_1^+(\ )$、$K_2^+(\ )$、$K_1^-(\ )$ 和 $K_2^-(\ )$ 表示加密/解密时所使用的 RSA 公钥和 RSA 私钥。

a. 使用一幅时序图，提供一个（尽可能简单的）协议允许 Alice 创建一个用于 Proxy1 的共享会话密钥 S_1。$S_1(m)$ 表示为使用共享密钥 S_1 对数据 m 加密/解密。
b. 使用时序图，提供一个（尽可能简单的）协议允许 Alice 创建一个对于 Proxy2 的共享会话密钥 S_2，而不向 Proxy2 透露她的 IP 地址。
c. 现在假设创建了共享密钥 S_1 和 S_2。使用时序图提供一个协议（尽可能简单并且不使用公开密钥密码），该协议允许 Alice 从 activist.com 请求一个 html 页面而不向 Proxy2 透露她的 IP 地址，并且不向 Proxy1 透露她正在访问哪个站点。你的图应当终止在一个 HTTP 请求到达 activist.com。

Wireshark 实验

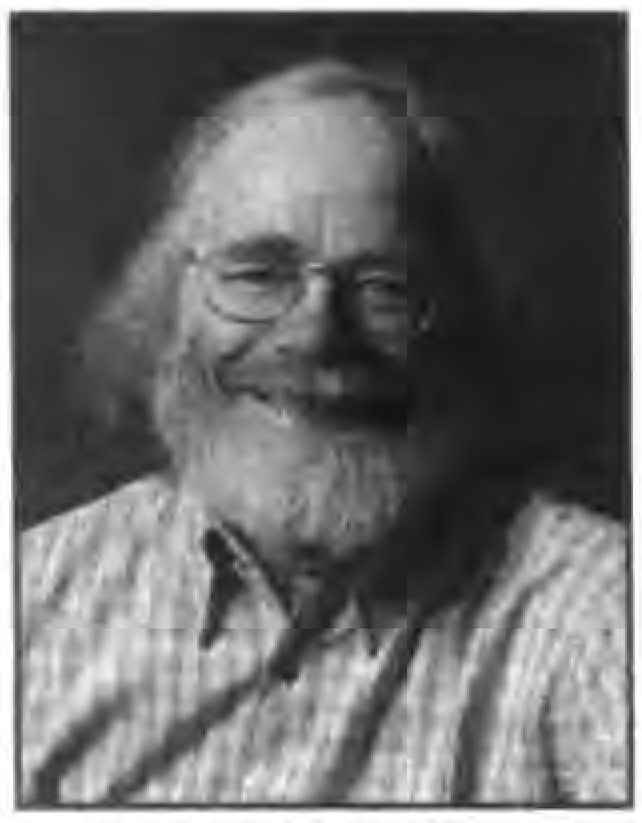
在这个实验中（与本书配套的 Web 站点有可用资源），我们研究安全套接层（SSL）协议。8.6 节讲过，使用 SSL 使得 TCP 连接更为安全，为了使因特网事务安全，实践中广泛应用了 SSL。在本实验中我们关注经 TCP 连接发送的 SSL 记录。我们将试图对每个记录定界和分类，目标是理解每个记录的工作原理和工作过程。我们研究各种 SSL 记录类型以及在 SSL 报文中的字段。通过分析你的主机与某电子商务服务器之间发送的 SSL 记录的踪迹来做这些事情。

IPsec 实验

在这个实验中（与本书配套的 Web 站点有可用资源），我们将探讨如何在 linux 装置之间创建 IPsec SA。你能够用两个普通的 linux 装置做该实验的第一部分，每个装置配有一块以太网适配器。但是对于实验

的第二部分，你将需要4个linux装置，这些装置每个都具有两块以太网适配器。在该实验的第二部分，你将在隧道模式中使用ESP协议创建IPsec SA。你做实验过程是：先人工创建SA，然后让IKE创建SA。

人物专访



Steven M. Bellovin

Steven M. Bellovin在位于新泽西州Florham Park的AT&T实验研究所的网络服务研究实验室工作多年后，成为哥伦比亚大学的教师。他的研究重点是网络和安全，以及将两者有机结合起来。1995年，因创立了Usenet，即第一个连接两个或多个计算机并允许用户共享信息和参与讨论的新闻组交换网络，而被授予Usenix终生成就奖。Steven也是国家工程学会的当选成员。他获得了哥伦比亚大学的学士学位和位于Chapel Hill的北卡罗来纳大学的博士学位。

- 什么原因使您决定专注于网络安全领域的研究？

听起来可能很奇怪，但是答案却很简单：只是因为感兴趣而已。我以前的背景是从事系统编程和系统管理，这很自然就发展到安全领域了。而且我一直对通信很感兴趣，这可以追溯到我还在上大学时，就兼职做系统编程方面的工作。

我在安全领域的工作持续受到两个因素的激励：一个是希望计算机有用，这意味着它们的功能不会被攻击者破坏，另一个是希望保护隐私。

- 当初您在研发Usenet时，您对它的愿景是什么？现在呢？

我们最初将它看作是一种能够在全国范围内讨论计算机科学和计算机编程的手段，考虑了用于事务管理和广告销售等目的的许多本地使用情况。事实上，我最初的预测是，每天从至多50~100个站点有1~2个报文。但是实际增长是与人相关的主题方面，包括（但不限于）人与计算机的相互作用。这么多年来，我喜欢的新闻组有rec. woodworking以及sci. crypt。

在某种程度上，网络新闻已经被Web取代。如果现在要我再设计它的话，就会和那时的设计大不相同了。但是它仍然是沟通对某一主题感兴趣的大量读者的一种极好手段，而不必依赖特定的Web站点。

- 是否有人给过您专业上的启示和灵感？以什么样的方式呢？

Fred Brooks教授对我的专业生涯影响重大。他是位于Chapel Hill的北卡罗来纳大学计算机科学系的创立者和原系主任，是研发IBM S/360和OS/360团队的管理者。他也是“The Mythical Man Mouth”（《人月神话》）的作者。最重要的是，他教给我们展望和折中的方法，即如何在现实世界环境中观察问题（不论这个现实世界比理论上的要复杂多少倍），以及在设计一种解决方案时如何平衡竞争各方的利益。大部分计算机工作都是工程性的，正确折中的艺术能够满足许多相矛盾的目标。

- 您对未来的联网和安全性的展望是什么？

到目前为止，我们所具有的安全性大多来自隔离。例如，防火墙的工作是通过切断某些机器和服务实现的。但是我们正处在增加连通性的时代，这使得隔离变得更为困难。更糟糕的是，我们的生产性系统要求的远不止是分离的部件，而需要通过网络将它们互联起来。我们面临的最大挑战之一是使所有都安全。

- 您认为在安全性方面已经取得的最大进展是什么？未来我们还能有多大作为？

至少从科学上讲，我们知道了密码学的原理。这是非常有帮助的。但是多数安全性问题因为其代码错误成堆而成为非常困难的问题。事实上，它是计算机科学中悬而未决的老问题，并且我认为该问题仍会持续。挑战在于弄明白：当我们不得不使用不安全的组件构建安全的系统时，如何才能让系统安全。我们面对硬件故障已经能够解决可靠性问题了；面对安全性问题，我们是否能够做到这一点呢？

- 您对进入因特网和网络安全领域的学生有什么忠告吗？

学习各种安全机制是件容易的事。学习如何“思维多疑”是困难的。你必须记住概率分布在下列场合并不适用，即攻击者能够发现不可能的情况。细节情况不胜枚举。

第9章

Computer Networking: A Top-Down Approach, Seventh Edition

多媒体网络

位于世界各个角落的人们懒散地靠坐在床上或乘坐公共汽车和地铁打发时间时，眼下都使用因特网来按需观看电影和电视节目。因特网电影和电视分发公司（如北美的 Netflix 和 Amazon、中国的“优酷”（Youku）和“看看”（Kankan））实际上已经成为家喻户晓的品牌。而人们不仅观看因特网视频，他们也使用诸如 YouTube 这样的站点来上载和分发用户自己生成的内容，不仅成为因特网视频的消费者，也成为视频的生产者。此外，网络应用如 Skype、Google Talk 和微信（WeChat 在中国极为流行），不仅允许人们经过因特网打“电话”，而且可以用视频和多方会议来强化这些电话。事实上，我们能够确定地预测：到当前年代末，几乎所有的视频消费和语音会话都将在因特网的端到端发生，通常更多是以无线终端方式经蜂窝和 WiFi 接入网与因特网相连接。传统的电话和广播电视正快速落伍。

本章以 9.1 节中的多媒体应用的分类方法开始。我们将看到多媒体应用能够分为*流式存储音频/视频*、*会话式 IP 音频/视频*或*流式实况音频/视频*等几类。我们将看到这些应用类型中的每一类都有自己独特的服务需求，这些需求与传统的弹性应用如电子邮件、Web 浏览和远程注册的需求差异很大。在 9.2 节中，我们较为详细地研究流式视频。我们将探讨支撑流式视频的许多基础原则，包括客户缓存、预取和对可用带宽的适应性视频质量。在 9.3 节中，我们研究会话式语音和视频。这种应用不同于弹性应用，对端到端时延高度敏感，但能够容忍偶尔的数据丢失。此时我们将研究诸如适应性播放、前向纠错和差错掩盖等技术是如何减缓网络引入的丢包和时延的。我们还将以学习案例方式审查 Skype。在 9.4 节中，我们将学习 RTP 和 SIP，这是两个用于实时会话式语音和视频应用的协议。在 9.5 节中，我们将研究网络内部的一些机制，这些机制能用于区分一类流量（如会话式语音这样的时延敏感应用）和其他类型流量（如浏览 Web 网页这样的弹性应用），并且在多类流量中提供区分服务。

9.1 多媒体网络应用

我们将多媒体网络应用定义为任何应用音频或视频的网络应用。在本节中，我们将提供多媒体应用的分类法。我们将看到在该分类法中的每类应用都具有自己独特的服务要求和设计问题集合。但在深入讨论因特网多媒体应用前，考虑音频和视频媒体自身的内在特点是有用的。

9.1.1 视频的性质

视频最为显著的特点或许是它的**高比特率**（high bit rate）。经因特网分发的视频的典型传输速率从用于低质量视频会议的 100kbps 到用于流式高分辨率电影的 3Mbps。为了比较视频带宽需求与其他因特网应用的带宽需求的不同，我们简要地考虑三个不同的用户，他们每人使用了一种不同的因特网应用。第一位用户 Frank，他打算迅速将照片张贴到他的朋友的脸书页面上。我们假设 Frank 每 10 秒查找一次新照片，并且这些照片的平均大小

是200KB。（与以前一样，我们在整个讨论中都简单地假定1KB=8000比特。）第二位用户Martha正从因特网（“云中”）向她的智能手机流式传输音乐。我们假定Martha正在听许多MP3歌曲，一首接着一首，都以128kbps速率进行编码。第三位用户Victor则正在观看以2Mbps编码的视频。最后，我们假设所有三位用户的会话长度是4000秒（大约67分钟）。表9-1比较了这三位用户的比特率和传输的总字节。我们看到这时流式视频消耗了最多的带宽，其比特率比脸书和流式音乐应用的带宽大10倍。因此，当设计网络视频应用时，我们心中必须记住的第一件事是视频的高比特率需求。鉴于视频的流行性及其高比特率，也许不会对思科公司的以下预测感到惊讶［Cisco 2015］：到了2019年，流式视频和存储视频将大约占全球因特网流量消费的90%。

表9-1 三种因特网应用的比特率需求的比较

	比特率	67分钟传输的字节
Frank 脸谱	160kbps	80MB
Martha 音乐	128kbps	64MB
Victor 视频	2Mbps	1GB

视频的另一种重要特点是它能被压缩，因而要在视频质量与比特率间进行折中。视频是一个图像序列，图像通常以恒定的速率显示，例如每秒24幅或30幅图像。一个没有压缩、数字编码的图像由像素阵列组成，每个像素被编码为一定数量的比特来表示亮度和颜色。在视频中有两种类型的冗余，它们都可以用来进行**视频压缩**（video compression）。空间冗余是给定图像的内部冗余。从直觉上讲，一个主要由空白组成的图像具有高度的冗余，能够有效地压缩而不会明显降低图像质量。时域冗余反映一幅图像和后续图像的重复程度。例如，如果一幅图像和后续图像完全一致，没有理由对后续图像再进行编码；相反，在编码过程中直接指出后续图像是完全一样的则更为有效。今天现成的压缩算法能够将视频压缩为所希望的任何基本比特率。当然，比特率越高，图像质量越好，总体用户视觉体验也越好。

我们也能够使用压缩来生成相同视频的**多重版本**（multiple version），每个版本有不同的质量等级。例如，我们能够使用压缩生成相同视频的三个版本，速率分别为300kbps、1Mbps和3Mbps。用户则能够根据他们的当前可用带宽来决定要观看哪个版本。具有高速因特网连接的用户可以选择3Mbps的版本；使用智能手机经过3G观看视频的用户可以选择300kbps的版本。类似地，在视频会议应用中的视频能被“动态”（on-the-fly）地压缩，以在会话用户之间给定的可用端到端带宽上提供最好的视频质量。

9.1.2 音频的性质

数字音频（包括数字化语音和音乐）的带宽需求比视频低得多。然而，数字音频具有自己独特的性质，当设计多媒体应用时必须考虑这些性质。为了理解这些性质，我们首先考虑模拟音频（由人和乐器所产生）是如何转换为数字信号的：

- 模拟音频信号首先以某种固定速率采样，例如每秒8000个样本。每个采样值是一个任意的实数。
- 然后每个采样值被“四舍五入”为有限个数值中的一个。这种操作被称为**量化**（quantization）。这些有限个数值（称为量化值）通常是2的幂，例如256个量化值。
- 每个量化值由固定数量的比特表示。例如，如果有256个量化值，那么每个值（因此每个音频采样）用一个字节来表示。所有样本的比特表示级联在一起就形成了该信号的数字表示。举例来说，如果一个模拟信号以每秒8000个样值采样，而且每个样本被量化并用8比特表示，则得到的数字信号的速率就为每秒64 000比

特。通过音频扬声器播放，这个数字信号则能够转换回来（也就是解码），形成一个模拟信号。然而，解码后的模拟信号仅是初始信号的近似，并且声音质量也许有明显的下降（例如，高频的声音可能在解码信号中丢失了）。通过增加采样速率和量化值的数量，解码信号能够更好地接近初始的模拟信号。因此（与视频一样），在解码信号的质量和比特率与数字信号存储空间之间存在一种折中。

我们刚才描述的基本编码技术称为**脉冲编码调制**（Pulse Code Modulation，PCM）。语音编码通常采用 PCM，采样速率为每秒 8000 个样本，每个样本用 8 比特表示，得到 64kbps 的速率。音频光盘（CD）也使用 PCM，采样速率为每秒 44 100 个样本，每个样本用 16 比特表示；这样使得单声道速率为 705.6kbps，立体声速率为 1.411Mbps。

然而，PCM 编码的语音和音乐很少在因特网中使用。与视频一样，取而代之的是使用压缩技术来减小流的比特速率。人类语音能被压缩到小于 10kbps 并仍然可懂。一种接近 CD 质量立体声音乐的流行压缩技术是 **MPEG 1 第 3 层**，更通常的叫法是 **MP3**。MP3 编码器通常能够压缩为许多不同的速率；128kbps 是最常使用的编码速率，并且能够产生非常小的声音失真。一种相关的标准是**高级音频编码**（Advanced Audio Coding，AAC），该标准已经随苹果公司而流行起来。与视频一样，能够以不同的比特率生成多重版本的预先录制的音频流。

尽管音频比特率通常比视频的比特率小得多，但用户通常对音频的小失误比视频的小失误更为敏感。例如，考虑在因特网上举行的视频会议。如果视频信号时不时地丢失几秒，该视频会议很可能继续进行而没有太多的用户抱怨。然而，如果音频信号经常丢失，用户就可能不得不中止该会话。

9.1.3 多媒体网络应用的类型

因特网能够支持各种各样有用的和娱乐性的多媒体应用。在本小节中，我们将多媒体应用分为三个大类：①流式存储音频/视频；②会话式 IP 语音/视频；③流式实况音频/视频。如我们很快将看到的那样，这些应用类型中的每种都有自己的服务需求和设计问题的集合。

1. 流式存储音频和视频

为使讨论具体化，我们这里聚焦流式存储视频，它通常结合了视频和音频组件。流式存储音频（例如流式音乐）非常类似于流式存储视频，尽管它的比特率通常要低得多。

在这类应用中，依赖的媒体是预先录制的视频（如电影、电视节目）预先录制的体育赛事或预先录制的用户生成的视频（如常在 YouTube 上看到的那些）。这些预先录制的视频放置在服务器上，用户向服务器发送请求按需观看视频。许多因特网公司今天提供流式视频，包括 YouTube（谷歌）、Netflix、Amazon 和 Hulu。流式存储视频具有三个关键的不同特色。

- *流*。在流式存储视频应用中，客户开始从服务器接收文件几秒之后，通常就开始播放视频。这意味着当客户正在从视频的一个位置开始播放时，与此同时正在从服务器接收该视频的后续部分。这种技术被称为**流**（streaming），它避免了在开始播放之前必须下载整个视频（并且引起一个潜在的长时延）。
- *相互作用*。因为媒体是预先录制的，用户可以对多媒体内容进行暂停、重新配置前进、重新配置倒退、快进等操作。从一个客户提出这种请求到该动作在客户端表现出来，可接受的响应时间应该小于几秒。

- 连续播放。一旦视频开始播放，它应该根据初始记录的时序进行。因此，为了在客户端播放，必须从服务器中及时接收数据；否则，用户经历视频帧停滞（这时客户等待延迟的帧）或帧跳过（这时客户漏掉延迟的帧）。

到目前为止，对流式视频最重要的性能测度是平均吞吐量。为了提供连续的播放，网络为流式应用提供的平均吞吐量必须至少与该流视频本身的比特率一样大。如我们将在9.2节所见，通过使用缓存和预取，即使在吞吐量波动的时候，提供连续播放也是可能的，只要平均吞吐量（在5~10秒区间平均）保持在视频速率之上［Wang 2008］。

对于许多流式视频应用，预先录制的视频被存储起来，并且从CDN而非从单一的数据中心流式播放。也有许多P2P视频流式应用，其中视频被存储在用户主机（对等方）上，不同视频块从可能分布在全球的不同对等方到达。在得知了因特网流式视频的性能后，我们将在9.2节更加深入地研究流式视频，特别关注客户缓存、预取、对可用带宽的适应性质量和CDN分发。

2. 会话式IP语音和视频

在因特网上的实时会话式语音通常称为**因特网电话**（Internet telephony），因为从用户的角度看，它类似于传统的电路交换电话服务。它也常被称为**IP语音**（Voice-over-IP，VoIP）。会话式视频与之类似，除了它包括参与者的语音以及视频外。今天的大多数语音和视频会话式系统允许用户生成具有三个或更多个参与者的会议。会话式语音和视频广泛地应用于今天的因特网中，因特网公司Skype、QQ和Google Talk自称每天都有数亿用户。

在第2章有关应用服务需求的讨论中（图2-4），我们确定了一些轴，服务需求可以根据它们分类。其中的两个轴（即定时考虑和数据丢失容忍度）对会话式语音和视频应用尤其重要。定时考虑是很重要的，因为音频和视频会话式应用是高度**时延敏感**（delay-sensitive）的。对于具有两个或更多个交互讲话者的会话来说，从用户讲话或移动开始到该动作显现在其他端的时延应当小于几百毫秒。对于语音，小于150ms的时延不会被人类听者觉察到，150~400ms的时延能够被接受，当时延超过400ms时，即使不会使对话变得完全无法理解，也会使语音会话变得令人沮丧。

另一个方面，会话式多媒体应用**容忍丢包**（loss-tolerant），即偶尔的丢失只会在音频/视频回放时偶尔出现干扰信号，而且这些丢失经常可以部分或者全部地隐藏。这些时延敏感但容忍丢包的特性明显不同于那些弹性数据应用（如Web浏览、电子邮件、社交网络和远程注册等）的特性。对于这些弹性应用，长时延令人烦恼，但并不是特别有害，然而传输数据的完全和完整性是首要的。我们将在9.3节中更加深入地探讨会话式语音和视频，特别关注适应性播放、前向纠错和差错掩盖是如何减缓网络引入的分组丢失和时延的。

3. 流式实况音频和视频

这种第三类应用类似于传统的电台广播和电视，只是它通过因特网来传输而已。这些应用允许用户接收从世界上任何角落发出的实况无线电广播和电视传输。今天有数以千计、遍及全球的无线电台和电视台正在因特网上广播内容。

实况是类似于广播的应用，它们经常有很多接收相同音频/视频节目的客户。在今天的因特网中，这种应用通常是用CDN来实现的（2.6节）。由于使用流式存储多媒体，网络必须为每个实况多媒体流提供大于该视频消耗速率的平均吞吐量。因为事件是直播的，尽管定时限制没有会话式语音那么严格，但时延也可能成为问题。从用户选择观看一个实况传输到播放开始，能够容忍的时延最多为10秒。我们在本书中将不涉及流式实况媒体，

因为用于流式实况媒体的许多技术（如初始缓存时延、适应性带宽使用和 CDN 分发）都类似于流式存储媒体所使用的技术。

9.2 流式存储视频

对于流式视频应用，预先录制的视频放置在服务器上，用户向这些服务器发送请求按需观看这些视频。用户可能从开始到结束都在观看视频而没有中断它，也可能在视频结束前停止观看它，或者通过暂停、重新定位到后面或前面镜头来与视频交互。流式视频系统可分为三种类型：**UDP 流**（UDP streaming）、**HTTP 流**（HTTP streaming）和**适应性 HTTP 流**（adaptive HTTP streaming）（参见 2.6 节）。尽管在实践中所有这三种系统都在使用，但绝大多数今天的系统应用了 HTTP 流和适应性 HTTP 流。

所有这三种形式的视频流的共同特点是广泛使用了客户端应用缓存，以此来缓解变化的端到端时延和变化的服务器和客户之间可用带宽量的影响。对于流式视频（存储的和实况的），用户通常能够容忍在客户请求某视频与该流视频在客户端播放之间有几秒的初始小时延。所以，当视频开始到达客户时，客户不必立即开始播放，反而能够在应用程序缓存中建立该视频的储备。一旦该客户建立起几秒“已缓存但尚未播放”的视频储备，客户就可以开始视频播放了。这种**客户缓存**（client buffering）具有两种重要的优点。第一，客户端缓存能够吸收服务器到客户时延中的波动。如果某特殊部分的视频数据延迟了，只要它在“接收到但尚未播放”的视频耗尽之前到达，这个长时延将不会被注意到。第二，如果服务器到客户带宽暂时低于视频消耗速率，用户能够继续享受连续的播放，只要客户应用缓存仍没有完全排尽。

图 9-1 显示了客户端的缓存。在这个例子中，假定视频以固定的比特率编码，因此每个视频块包含了能在相同固定时间量 Δ 区间播放的视频帧。服务器在 t_0 传输第一个视频块，在 $t_0+\Delta$ 传输第二个视频块，在 $t_0+2\Delta$ 传输第三个视频块等等。一旦客户开始播放，为了重新产生初始录制视频的定时，每个块应当在前一个块之后播放 Δ 时间单元。第一个视频块于 t_1 时刻到达，第二个视频块于 t_2 时刻到达。第 i 块的网络时延是服务器传输该块

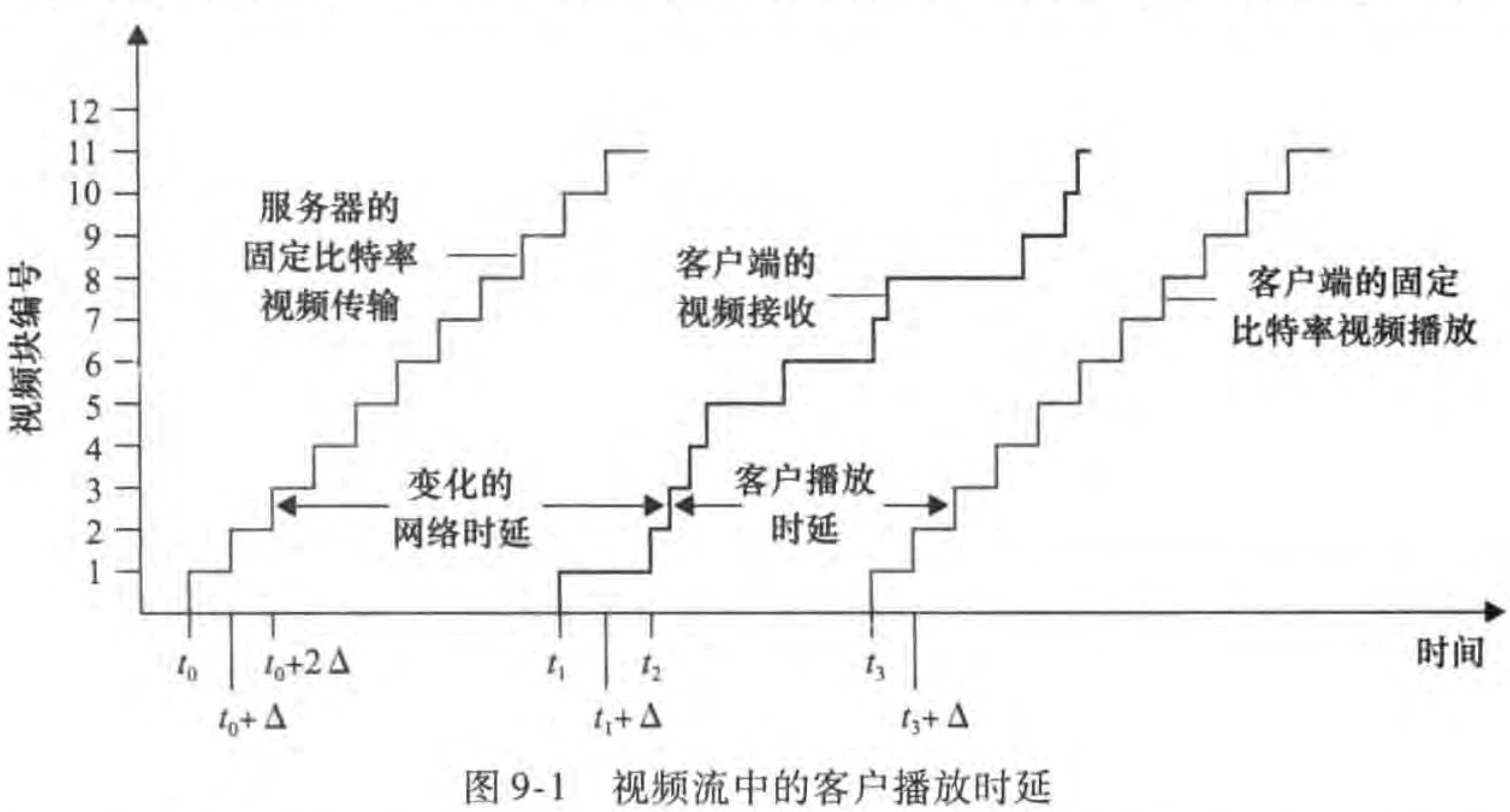


图 9-1 视频流中的客户播放时延

的时间与客户收到该块的时间之间的水平距离；注意到网络时延随视频块不同而变化。在此例子中，如果客户准备当第一块在 t_1 时刻一到达就开始播放，那么第二块将不能在 $t_1+\Delta$时刻及时到达进行播放。在这种情况下，视频播放或将停止运行（等待第二块的到

达）或可能漏掉第二块，即这两种情况都将导致不希望的播放损伤。相反，如果客户将播放延迟到 t_3 时刻开始，这时第一块到第六块都已经到达，所有已经收到的块在它们的播放时间前都能够进行周期性的播放。

9.2.1 UDP 流

我们这里仅简要讨论 UDP 流，在适当时向读者更为深入地介绍这些系统背后隐含的协议。使用 UDP 流，服务器通过 UDP 以一种稳定的速率记录下视频块，用与客户的视频消耗速率相匹配的速率传输视频。例如，如果视频消耗率是 2Mbps，每个 UDP 分组承载 8000 比特视频，则服务器将每隔（8000 比特）/（2Mbps）=4ms 向其套接字发送一个 UDP 分组。如我们在第 3 章所知，因为 UDP 未采用某种拥塞控制机制，所以服务器能够以视频的消耗速率将分组推进网络中，而无 TCP 的速率控制的限制。UDP 流通常使用很小的客户端缓存，空间维持小于 1 秒视频就足够了。

在将视频块传递给 UDP 之前，服务器将视频块封装在运输分组中，该运输分组是专门为传输音频和视频而设计的，使用了**实时传输协议**（Real-Time Transport Protocol，RTP）[RFC 3550] 或某种类似（可能是专用）的方案。我们将在 9.3 节再讨论 RTP，那时我们将在会话式语音和视频系统环境中讨论 RTP。

UDP 流的另一种不同的性质是，除了服务器到客户的视频流外，两者间还并行地维护一个单独的控制连接，通过该连接，客户可发送有关会话状态变化的命令（如暂停、重新开始、重定位等）。这种控制连接在许多方面类似于我们在第 2 章中学习的 FTP 控制连接。在本书配套 Web 网站上更为详细地解释了**实时流协议**（Real-Time Streaming Protocol，RTSP）[RFC 2326]，它是一种用于这样的控制连接的流行开放协议。

尽管 UDP 流已经在多个开源系统和专用产品中得到应用，但它有三个重大不足。首先，由于服务器和控制之间的可用带宽无法预测并且是变化的，恒定速率 UDP 流不能够提供连续的播放。例如考虑以下场景：视频消耗速率为 1Mbps，服务器到客户可用带宽通常超过 1Mbps，但每过几分钟就有几秒时间其可用带宽低于 1Mbps。在这种场景下，以 1Mbps 恒定速率经 RTP/UDP 传输视频的 UDP 流系统很可能将提供不好的用户体验，在可用带宽低于 1Mbps 之后产生停滞或漏帧。UDP 流的第二个缺点是它要求如 RTSP 服务器这样的媒体控制服务器，以对每个进行中的客户会话处理客户到服务器的交互请求和跟踪客户状态（例如在视频中的客户播放点，视频是否被暂停或播放等）。这增加了部署大规模的按需视频系统的总体成本和复杂性。第三个缺点是许多防火墙配置为阻塞 UDP 流量，防止这些防火墙后面的用户接收 UDP 视频。

9.2.2 HTTP 流

在 HTTP 流中，视频直接作为具有一个特定 URL 的普通文件存储在 HTTP 服务器上。当用户要看视频时，客户和服务器之间建立一个 TCP 连接，并且发送一个对该 URL 的 HTTP GET 请求。服务器则尽可能快地在 HTTP 响应报文中发送该视频文件，这就是说，以 TCP 拥塞控制和流控制允许的尽可能快的速率进行处理。在客户端上，字节收集在一个客户应用缓存中。一旦在缓存中字节数量超过了预先设定的阈值，该客户应用程序开始播放，具体而言，它周期性地从客户应用缓存中抓取视频帧，对帧解压缩并在用户屏幕上显示它们。

我们在第 3 章学习过，当通过 TCP 传输一个文件时，由于 TCP 的拥塞控制机制，服

务器到客户的传输速率可能变化很大。特别是，传输速率以与 TCP 拥塞控制相关联的“锯齿”形变化并非罕见。此外，分组也能由于重传机制而被大大延迟。因为 TCP 的这些特点，在 20 世纪 90 年代大多数人关于会话式的看法是流式视频将不可能在 TCP 上很好地工作。然而，随着时间的推移，流式视频系统的设计者知道了当使用了客户缓存和预取（在下面讨论）技术时，TCP 的拥塞控制和可靠数据传输机制并不一定会妨碍连续播放。

在 TCP 上使用 HTTP 也使得视频穿越防火墙和 NAT（它们常常被配置为阻挡 UDP 流量但允许大部分 HTTP 流量通过）更为容易。HTTP 流消除了因需要媒体控制服务器（如 RTSP 服务器）带来的不便，减少了在因特网上大规模部署的成本。由于所有这些优点，今天的大多数流式视频应用（包括 YouTube 和 Netflix）都使用 HTTP 流（在 TCP 上）作为它的底层流式协议。

1. 预取视频

如同我们刚才所学的那样，客户端缓存可用于缓解变化的端到端时延和变化的可用带宽的影响。在前面图 9-1 的例子中，服务器以视频播放的速率传输。然而，对于流式存储视频，客户能够尝试以高于消耗速率的速率下载视频，因此**预取**（prefetching）将来会被消耗的视频帧。该预取的视频当然存储在客户应用缓存中。这样的预取自然伴随 TCP 流出现，因为 TCP 拥塞避免机制将试图使用服务器和客户之间的所有可用带宽。

为了深入洞察预取技术，我们来举个简单的例子。假设视频消耗速率是 1Mbps，而网络从服务器到客户能够以恒定的 1.5Mbps 速率交付视频。客户则不仅能够以非常小的播放时延播放该视频，而且还能够以每秒 500Kb 的量增加缓存的视频数据。以这种方式，如果后来该客户在一段短暂时间内以小于 1Mbps 的速率接收数据，该客户由于在其缓存中的储备将能够继续提供连续的播放。[Wang 2008] 显示了当平均 TCP 吞吐量大致为媒体比特率的两倍时，TCP 流导致最小的饥饿和低缓存时延。

2. 客户应用缓存和 TCP 缓存

图 9-2 说明了客户和服务器之间 HTTP 流的交互。在服务器侧，视频文件中的白色部分已经通过服务器的套接字进行发送，而黑色部分是留下待发送的部分。在“通过套接字的门传送”之后，放置在 TCP 发送缓存中的字节在被传输进因特网之前如第 3 章所描述。在图 9-2 中，因为 TCP 发送缓存显示为满，服务器立即防止从视频文件向套接字发送更多的字节。在客户侧，客户应用程序（媒体播放器）从 TCP 接收缓存（通过其客户套接字）读出字节并将字节放入客户应用缓存中。与此同时，客户应用程序周期性地从客户应用缓存中抓取视频帧，解压缩并显示在用户屏幕上。注意到如果客户应用缓存大于该视频文件，则从服务器存储器到客户应用缓存移动字节的整个过程等价于普通文件经 HTTP 的下载过程，即客户直接将视频用 TCP 允许的尽可能快的速率从服务器中拉出来。

现在考虑在流播放期间当用户暂停视频时将发生的现象。在暂停期间，比特未从客户应用缓存中删除，甚至比特继续从服务器进入缓存。如果客户应用缓存是有限的，它可能最终会变满，这将反过来引起对服务器的“反向压力”。具体而言，一旦客户应用缓存变满，字节不再从客户 TCP 接收缓存中删除，因此它也会变满。一旦客户 TCP 接收缓存变满，字节不再从服务器 TCP 发送缓存删除，因此它也变满。一旦客户 TCP 发送缓存变满，服务器不能向套接字中发送任何更多的字节。因此，如果用户暂停视频，服务器可能被迫停止传输，在这种情况下服务器被阻塞，直到用户恢复该视频。

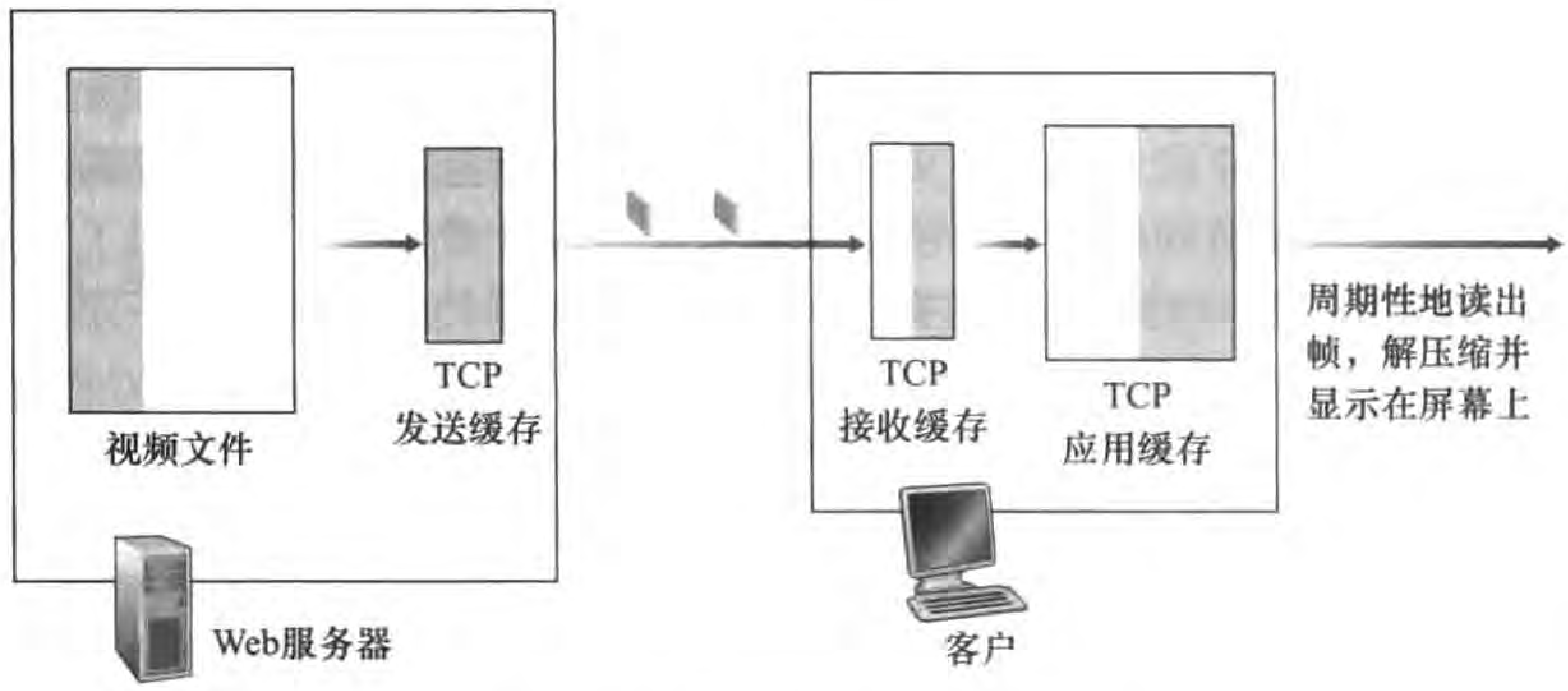


图9-2　经HTTP/TCP的流式存储视频

事实上，甚至在常规的播放过程中（即没有暂停），如果客户应用缓存变满，反向压力将引起TCP缓存变满，这将迫使服务器降低其速率。为了决定其产生的速率，注意到当客户缓存删除f比特，它在客户应用缓存中产生了f比特的空间，这依次允许服务器发送额外的f比特。因此，服务器发送速率不能比客户端视频消耗速率更高。因此，当使用HTTP流时，一个满的客户应用缓存间接地对服务器到客户能够发送的视频速率施加了限制。

3. 流式视频的分析

某些简单的建模将有助于洞察由于应用缓存消耗所产生的初始播放时延和停滞。如图9-3所示，B表示客户应用缓存的长度（以比特计），Q表示在客户应用缓存开始播放之前必须被缓存的比特数量。（当然，$Q<B$。）r表示视频消耗速率，即客户在播放期间从客户应用缓存提取比特的速率。在此情况下，举例来说，如果视频的帧速率是30帧/秒，每（压缩）帧是100 000比特，则$r=3\text{Mbps}$。为了从细节看整体，我们将忽略TCP的发送和接收缓存。

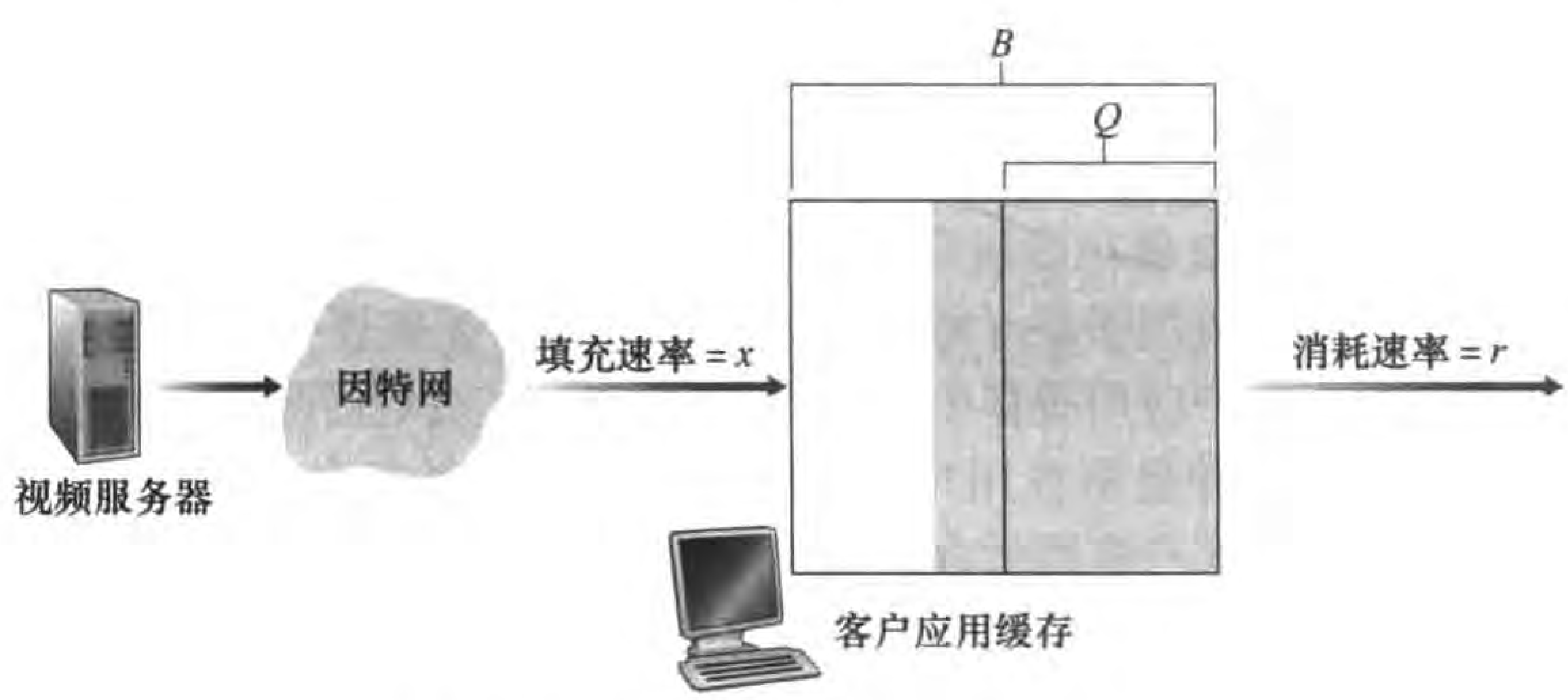


图9-3　流式视频的客户端缓存的分析

我们假设无论何时客户缓存都为非空，服务器以一种恒定速率x发送比特。（这是一种显而易见的简化，因为TCP的发送速率由于拥塞控制而变化；在本章后面的习题中我们将考察更为真实的与时间相关的速率$x(t)$。）假设在时刻$t=0$，应用缓存为空，视频开始到达客户应用缓存。我们现在问，在什么时刻$t=t_p$开始播放呢？并且在播放过程中，什么时刻$t=t_f$客户应用缓存变满呢？

首先，我们来确定 t_p，此时 Q 比特已经进入应用缓存并且开始播放。前面讲过比特以速率 x 到达客户应用缓存，并且在开始播放之前没有比特从其缓存中删除。所以，建立 Q 比特所需的时间（初始缓存时延）是 $t_p = Q/x$。

我们现在来决定 t_f，这是客户应用缓存变满的时刻。我们先观察，如果 $x<r$（即如果服务器发送速率小于视频消耗速率），则客户缓存将决不会满！的确，时刻 t_p 开始，缓存将以速率 r 排空并且仅以速率 $x<r$ 填充。最终客户缓存将完全排空，此时当客户缓存等待另一个 t_p 秒来建起 Q 比特的视频时，视频将在屏幕上停滞。*所以，当网络中可用速率小于视频速率时，播放将在连续播放期和停滞播放期之间进行变动。*在课后习题中，将请你决定每个连续播放期和停滞期的长度，它们都作为 Q、r 和 x 的函数。当 $x>r$ 时，现在我们来决定 t_f。在这种情况下，在时刻 t_p 开始，缓存以 $x-r$ 的速率从 Q 增加到 B，因为比特以速率 r 消耗但以速率 x 到达，如图 9-3 所示。有了这些提示，在课后习题中将请你决定 t_f，即客户缓存变满的时刻。注意到*当网络中的可用速率大于视频速率时，在初始缓存时延后，用户将享受连续的播放直到视频结束。*

4. 视频的早期中止和重定位

HTTP 流系统经常利用 HTTP GET 请求报文中的 **HTTP 字节范围首部**（HTTP byte-range header），该首部指示了客户当前要从所希望的视频中获取的字节范围。当用户要在视频中及时重定位（即跳跃）到未来点时，这特别有用。当用户重定位到一个新位置时，客户发送一个新 HTTP 请求，用字节范围首部指出服务器应当从文件的哪个字节起发送数据。当服务器接收到该新的 HTTP 请求时，它能够忘记任何较早的请求，而是由字节范围请求中指示的字节开始发送。

在我们讨论重定位主题的时候，我们简要地提及当某用户重定位到视频中的某个未来点或提前终止视频时，某些由服务器发送的已预取但尚未观看的数据将不会被观看，即导致了网络带宽和服务器资源的浪费。例如，假设在视频中的某时刻 t_0 客户缓存充满 B 比特，在此时用户重定位到视频中的某个瞬间 $t>t_0+B/r$，然后从这点起观察视频直到结束。在这种情况下，缓存中的所有 B 比特将未被观看，用于传输这 B 比特的带宽和服务器资源完全被浪费掉了。在因特网中，有大量的带宽因提前终止而浪费，这些成本可能相当大，特别是对于无线链路 [Ihm 2011]。由于这个原因，许多流系统仅使用了长度适当的客户应用缓存，或者将限制在 HTTP 请求中使用字节范围首部预取的视频数量 [Rao 2011]。

重定位和提前终止可以与下列做法类比：烹调了很多肉，仅吃了一部分，并将其他都扔掉，因而浪费了食物。因此，下次你因为没有吃完所有晚餐而被父母批评浪费食物时，你能够迅速反驳：当他们在因特网上观看电影并进行带宽重定位时，他们浪费了带宽和服务器资源！但是，别人错了不等于你对了，食物和带宽都不应被浪费！

在 9.2.1 节和 9.2.2 节中，我们分别学习了 UDP 流和 HTTP 流。第三种流是经 HTTP 的动态适应流（DASH），DASH 使用了该视频的多个版本，每个版本是以不同速率压缩而成的。DASH 在 2.6.2 节中进行了详细讨论。分发存储和实况视频经常使用 CDN。CDN 在 2.6.3 节中进行了详细讨论。

9.3 IP 语音

经因特网的实时会话式语音经常被称为**因特网电话**（Internet telephony），因为从用户的视角看，它类似于传统的电路交换电话服务。它通常也被称为 **IP 语音**（Voice-over-IP,

VoIP)。在本节中我们描述 VoIP 所依据的原则和协议。会话式视频在许多方面类似于 VoIP，除了它包括参与者的视频以及他们的语音以外。为了使讨论重点突出且具体，我们这里仅关注语音，而不是语音和视频的结合。

9.3.1 尽力而为服务的限制

因特网的网络层协议 IP 提供了尽力而为的服务。那就是说服务尽全力将每个数据报从源尽可能快地移动到目的地，但是没有就在某些时延界限内使分组到达目的地或丢包百分比的限制做任何承诺。缺失这种保证对实时会话式应用的设计提出了严峻挑战，这些应用对分组时延、时延抖动和丢包非常敏感。

在本节中，我们将讨论加强尽力而为网络上的 VoIP 性能的几种方式。我们的重点将在应用层技术上，即这些技术并不要求在网络核心甚至在端系统的运输层有任何变化。为了使讨论具体，我们将讨论在一个特定的 VoIP 例子环境下尽力而为 IP 服务的限制。发送方以每秒 8000 字节的速率产生字节，且每 20ms 将字节汇聚成块。每个块和一个特殊的首部（在下面讨论）封装在一个 UDP 报文段中（通过一个到套接字接口的呼叫）。因此，一个块中的字节数为（20ms）×(8000 字节/秒) = 160 字节，每 20ms 发送一个 UDP 报文段。

如果每个分组以恒定的端到端时延到达接收方，那么分组每隔 20ms 就能周期性地到达接收方。在这种理想的情况下，只要每个块一到达，接收方就能直接播放它。但不幸的是，某些分组可能丢失，大多数分组没有相同的端到端时延，即使在一个轻度拥塞的因特网中也是如此。因此，接收方必须更仔细地判断：①什么时候播放一个块；②如何处理一个丢失块。

1. 丢包

考虑由 VoIP 应用产生的一个 UDP 报文段。这个 UDP 报文段封装在 IP 数据报中。当数据报在网络中徘徊时，在它等待出链路传输时要经过路由器的缓存（即队列）。从发送方到接收方的路径上的一个或多个缓存有可能是满的，不能接纳该 IP 数据报。在这种情况下，这个 IP 数据报就被丢弃了，永远不会到达接收方的应用程序。

通过 TCP（它提供了可靠数据传输）而不是 UDP 发送分组可以消除丢失。然而，重传机制对于诸如 VoIP 这样的会话式实时音频应用，通常认为是不可接受的，因为它们增加了端到端时延［Bolot 1996］。此外，当丢包后，由于 TCP 的拥塞控制，发送方的传输速率可能减少到低于接收方的排空速率，可能导致缓存“饥饿”。这可能会对接收方的语音可理解程度产生严重影响。由于这些原因，几乎所有现有的 VoIP 应用默认运行在 UDP 上。［Baset 2006］报告称 Skype 使用了 UDP，除非用户位于阻碍 UDP 报文段的 NAT 或防火墙之后（这时使用 TCP)。

但是分组的丢失并不一定会造成人们想象中的灾难。实际上，取决于语音是如何编码和传输的以及接收方隐藏丢包的方式，1% ~20% 的丢包率是可以忍受的。例如，前向纠错（FEC）能够有助于隐藏丢包。我们后面可以看到，通过使用 FEC，将冗余信息和初始信息一起传输，以便能够从冗余信息中恢复一些丢失的初始数据。无论如何，如果发送方和接收方之间的一段或多段链路严重拥塞，丢包率超过 10% ~20%（例如在无线链路上），那么无论采取何种措施都无法获得可以接受的声音质量了。显然，尽力而为服务有它的局限性。

2. 端到端时延

端到端时延（end-to-end delay）是以下因素的总和：路由器中的传输、处理和排队时

延，链路中的传播时延和端系统的处理时延。对于实时会话式应用，例如VoIP，听电话的人对于小于150ms的端到端时延是觉察不到的；在150ms和400ms之间的时延能够接受，但是不够理想；超过400ms的时延可能严重妨碍语音谈话的交互性。VoIP应用程序的接收方通常忽略时延超过特定阈值（例如超过400ms）的任何分组。因此，时延超过该阈值的分组等效于丢弃。

3. 分组时延抖动

端到端时延的一个关键成分是一个分组在网络路由器中经历的变化的排队时延。由于这些可变的时延，从在源中产生分组到它在接收方收到的这段时间，对于不同的分组可能会有波动，如图9-1所示。这个现象称为**时延抖动**（jitter）。举一个例子，考虑在VoIP应用中两个连续的分组。发送方在发送第一个分组20ms之后发送第二个分组。但是在接收方，这两个分组之间的间隔可能变得大于20ms。为了理解这一点，假设第一个分组到达路由器的一个几乎为空的队列，但是恰好在第二个分组到达该队列之前，从其他源来的大量分组也到达了相同的队列。因为在这个路由器中第一个分组经受了很小的排队时延，而第二个分组经受了较大的排队时延，这两个连续的分组之间的时间间隔变得大于20ms了。两个连续分组的间隔也可能会小于20ms。为了理解这种情况，再次考虑两个连续的分组。假设第一个分组加入一个有大量分组队列的队尾，并且第二个分组到达这个队列时第一个分组尚未被传输，而且来自其他源的分组尚未到达该队列。在这种情况下，这两个分组发现它们在队列中互相紧挨着。如果在路由器的出链路传输一个分组所需时间小于20ms，则第一个分组和第二个分组的间隔就变得小于20ms了。

这种情况可以与在公路上开车相类比。假设你和你的朋友每人驾驶一辆车从美国的圣地亚哥到凤凰城。并且假设你和朋友有类似的驾驶风格，而且你们都以交通规则允许的100km/h的速度驾驶。如果你的朋友在你之前一个小时出发，那么根据干扰车流量的不同，你可能在你的朋友之后一个小时左右到达凤凰城。

如果接收方忽略了时延抖动的存在，一旦该块到达就开始播放，那么在接收方产生的音频质量很容易变得不可理解。幸运的是，时延抖动通常可以通过使用**序号**（sequence number）、**时间戳**（timestamp）和**播放时延**（playout delay）来消除，如下面所讨论的内容。

9.3.2 在接收方消除音频的时延抖动

对于VoIP应用，周期性地产生分组，接收方应该在存在随机网络时延抖动的情况下尝试提供播放语音块。这经常通过结合下面两种机制来实现：

- 为每个块预先计划一个**时间戳**（timestamp）。发送方用每个块产生的时刻为它加上时间印记。
- 在接收方**延迟播放**（delaying playout）块。如我们前面在图9-1的讨论所见，接收的音频块的播放时延必须足够长，以便大多数分组在它们的预定播放时间之前被接收到。这个播放时延可能在整个音频会话期间是固定的，或者在音频会话生命期中适应性地变化。

我们现在讨论如何结合这三种机制来减轻甚至消除时延抖动的影响。我们研究两种播放策略：固定播放时延和适应性播放时延。

1. 固定播放时延

使用固定播放时延策略，接收方试图在块产生正好q ms后播放它。因此如果一个块

在时刻 t 打上时间戳，接收方在时刻 $t+q$ 播放这个块，假设这个块在那个时间已经到达。在预定播放时间之后到达的分组将被丢弃，并被认为已经丢失。

q 选择什么值为好呢？尽管使用更小的 q 值可以获得更令人满意的会话体验，但 VoIP 能够支持高达约 400ms 的时延。另一方面，如果 q 比 400ms 小得多，那么由于网络引入的分组时延抖动会使许多分组可能错过了它们的预定播放时间。概括地说，如果端到端时延经常发生大的变化，用一个大的 q 更好；另一方面，如果时延很小并且时延变化也很小，用一个较小的、可能小于 150ms 的 q 更好。

在图 9-4 中说明了播放时延和丢包之间的折中。该图表示了单个话音突峰期的分组产生和播放的时间。考虑了两种不同的初始播放时延。如最左边的阶梯所示，发送方以规则的间隔（比方说每 20ms）产生一组分组。在这个话音突峰期中的第一个分组在时刻 r 被接收到。如该图所示，由于网络时延抖动，后续分组的到达间隔是不均匀的。

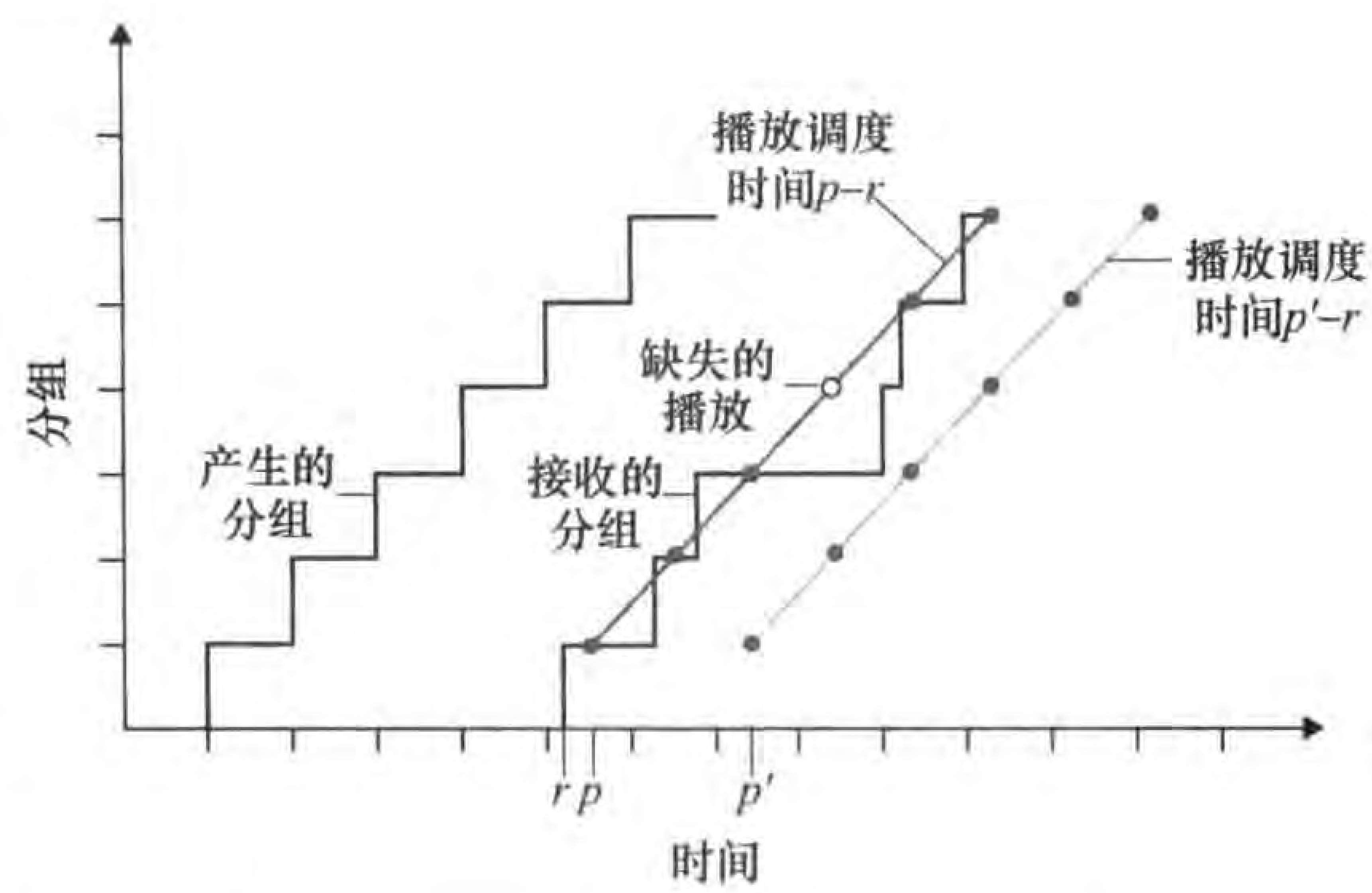


图 9-4 不同的固定播放时延情况下的丢包

对于第一个播放调度时间，固定的初始播放时延设置为 $p-r$。使用这个方案，第四个分组没有在它调度的播放时间到达，接收方认为它丢失了。对于第二个调度时间，固定的初始播放时延设置为 $p'-r$。对于这个方案，所有分组都在它们调度的播放时间之前到达，因此没有丢失。

2. 适应性播放时延

上面的例子显示了当使用固定播放时延来设计播放策略时所引起的重要的时延与丢包折中。若初始播放时延设置得比较大，大多数分组能在它们的截止时间内到达，因此存在的丢失将可忽略不计；然而，对于如 VoIP 这样的会话式服务，长时延即使不是不能忍受的，至少也是令人厌恶的。在理想情况下，我们希望播放时延最小化，使丢包低于一定百分比的限制。

处理这种折中的自然方法是估计网络时延和网络时延变化，并且在每个话音突峰期的开始相应地调整播放时延。在话音突峰期开始时适应性地调整播放时延将导致发送方静默期的压缩和拉长；然而，静默期的少量压缩和拉长在谈话中是不易觉察的。

根据［Ramjee 1994］，我们现在描述一种接收方可以用于适应性地调整它的播放时延的通用算法。为此，令

t_i = 第 i 个分组的时间戳 = 该分组在发送方产生的时间

r_i = 分组 i 被接收方接收的时间

p_i = 分组 i 在接收方播放的时间

第 i 个分组的端到端网络时延是 r_i-t_i。由于网络时延抖动，这个时延在不同的分组之间会发生变化。令 d_i 表示接收到第 i 个分组时的平均网络时延的估计值。这个估计值根据如下的时间戳来构造：

$$d_i = (1-u)d_{i-1} + u(r_i - t_i)$$

式中 u 是一个固定的常数（例如，$u=0.01$）。这样 d_i 是观察到的网络时延 r_1-t_1，…，r_i-t_i的一个平滑均值。这个估计值为最近观察到的网络时延设置了比过去一段时间观察到的网络时延有更大的权重。这种估值的形式不应该是完全陌生的；相似的思想在第 3 章讨论估计 TCP 往返时间时就使用过。令 v_i 表示与估计平均时延的平均时延绝对偏差的估计值。这个估计值也可从这些时间戳构建：

$$v_i = (1-u)v_{i-1} + u\,|r_i - t_i - d_i|$$

为每个接收的分组计算估计值 d_i 和 v_i，尽管它们仅能用于为任何话音突峰期的第一个分组确定播放点。

一旦计算完了这些估计值，接收方为分组播放应用下列的算法。如果分组 i 是一个话音突峰期的第一个分组，它的播放时间 p_i 计算如下：

$$p_i = t_i + d_i + Kv_i$$

这里 K 是一个正的常数（例如 $K=4$）。Kv_i 项的目的是给将来设置足够大的播放时间，以便话音突峰期中只有一小部分到达的分组由于迟到而丢失。在一个话音突峰期中任何后续分组的播放点被计算为对于这个话音突峰期的第一个分组播放时间点的偏移。特别是，令

$$q_i = p_i - t_i$$

表示从话音突峰期的第一个分组产生到它播放的时间长度。如果分组 j 也属于这个话音突峰期，它播放的时刻是

$$p_j = t_j + q_i$$

刚才描述的算法在假设接收方能够辨明一个分组是否是话音突峰期中的第一个分组的情况下很有意义。这能够通过检查每个接收到的分组中的信号能量来做到。

9.3.3 从丢包中恢复

我们已经较为详细地讨论了一个 VoIP 应用能够怎样处理分组时延抖动。我们现在简要地描述在存在丢包的情况下几种试图保护可接受的音频质量的方案。这样的方案被称为**丢包恢复方案**（loss recovery scheme）。这里我们定义了广义的丢包：如果某分组不能到达接收方或者在它调度的播放时间之后才到达，该分组则丢失。我们再次用 VoIP 例子作为描述丢包恢复方案的环境。

如在本节开始提到的那样，在诸如 VoIP 等会话式实时应用中，重传丢失的分组通常是不可行的。的确，重传一个已经错过了播放截止时间的分组是绝对没有意义的。而且重传一个在路由器队列溢出的分组通常不能足够快地完成。由于这些考虑，VoIP 应用通常使用某种类型的丢包预期方案。两种类型的丢包预期方案是**前向纠错**（Forward Error Correction，FEC）与**交织**（interleaving）。

1. 前向纠错

FEC 的基本思想是给初始的分组流增加冗余信息。以稍微增加传输速率为代价，这些冗余信息可以用来重建一些丢失分组的近似或者准确版本。根据文献［Bolot 1996］和［Perkins 1998］，我们现在概括了两种简单的 FEC 机制。第一种机制是每发送 n 个块之后发送一个冗余编码的块。这个冗余块通过异或 n 个初始块来获得［Shacham 1990］。以这种方式，在这 $n+1$ 个分组的组中，如果任何一个分组丢失，接收方能够完全重建丢失的分组。但是如果这一组中有两个或更多分组丢失，接收方则无法重建丢失的分组。通过让

组的长度 $n+1$ 比较小，当丢失不是很多时，大部分丢失分组都可以恢复。然而组的长度越小，相对增加的传输速率就越大。特别是若传输速率以 $1/n$ 因子增加的话，例如 $n=3$，则传输速率增加33%。此外，这个简单的方案增加了播放时延，因为接收方在能够开始播放之前，必须等待收到整个组的分组。有关 FEC 在多媒体传输上如何工作的更多实践细节参见［RFC 5109］。

第二个 FEC 机制是发送一个较低分辨率的音频流作为冗余信息。例如，发送方可能创建一个标称的音频流和一个相应的低分辨率、低比特率的音频流。（这个标称流可能是一个 64kbps 的 PCM 编码，而这个较低质量的流可能是一个 13kbps 的 GSM 编码。）这个低比特率流被认为是冗余信息。如图 9-5 所示，发送方通过从这个标称流中取出第 n 个块并附加上第（$n-1$）个块的冗余信息，以构建第 n 个分组。以这种方式，只要没有连续分组的丢失，接收方都可以通过播放和后续分组一起到达的低比特率编码块来隐藏丢失。当然，低比特率块比标称块的质量要低。然而，在一个流主要是由高质量块组成、偶尔出现低质量块并且没有丢失的块的情况下，其整体的音频质量良好。注意到在这种方案中，接收方在播放前只需接收两个分组，因此增加的时延小。此外，如果低比特率编码比标称编码少得多，那么传输速率的额外增加并不大。

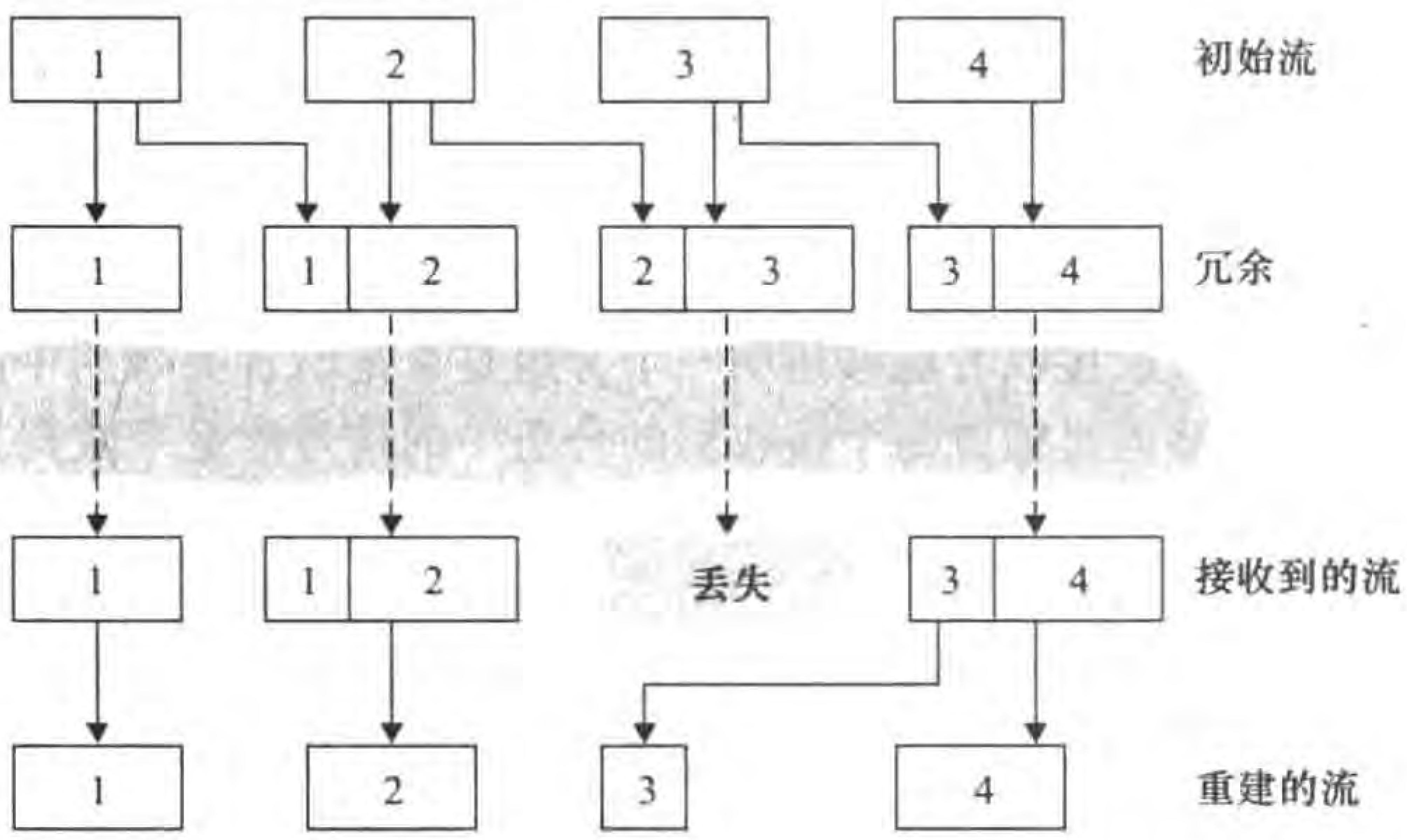


图 9-5　捎带低质量的冗余信息

为了处理连续丢失，我们能够使用率一个简单的修正方案。发送方不再仅为第 n 个标称块附加上第（$n-1$）个低比特率块，而是附加上第（$n-1$）个和第（$n-2$）个低比特率块，或者附加第（$n-1$）个和第（$n-3$）个低比特率块等等。通过给每个标称块附加上更多低比特率块，在各种恶劣的尽力而为服务环境下接收方的音频质量变得可接受。在另一方面，附加的块增加了传输带宽和播放时延。

2. 交织

作为冗余传输的另一种替代方案，VoIP 应用可以发送交织的音频。如图 9-6 所示，发送方在传输之前对音频数据单元重新排序，使得最初相邻的单元在传输流中以一定距离分离开来。交织可以减轻丢包的影响。例如，如果每个单元长为 5ms，块是 20ms（也就是每个块 4 个单元），那么第一个块可能包含 1、5、9 和 13 单元；第二个块可能包含 2、6、10 和 14 单元等等。图 9-6 显示了一个交织流的单个丢包导致重建流中的多个小间隙，这与在非交织流中将会导致单个大间隙形成对照。

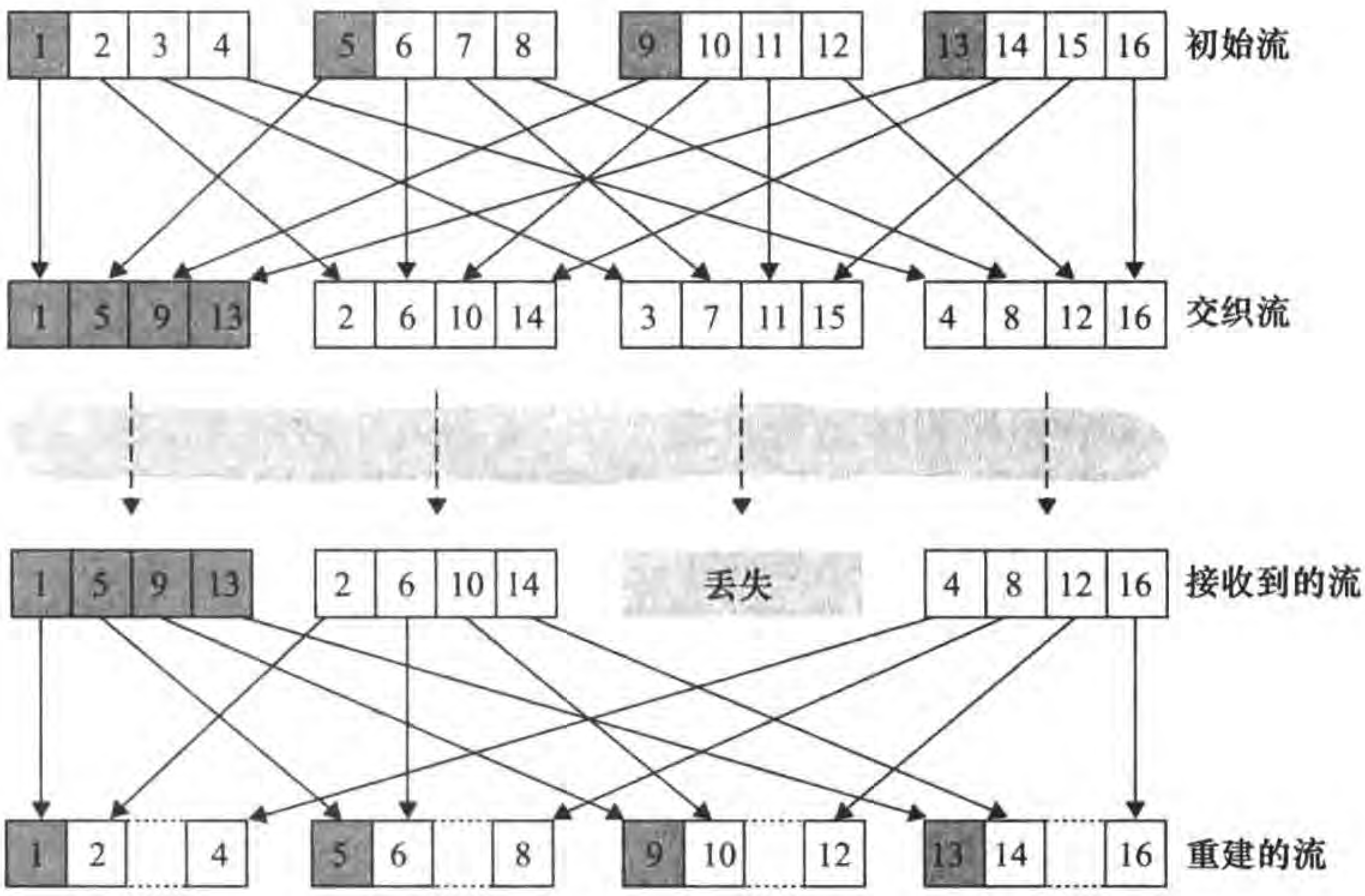


图 9-6 发送交织音频

交织能够明显地提高音频流可感觉到的质量 [Perkins 1998]。它的开销也较低。交织明显的缺点是增加了时延。这限制了它在如 VoIP 这样的会话式应用中的使用，然而它能够很好地处理流式存储音频。交织的一个主要优点是它不增加流的带宽需求。

3. 差错掩盖

差错掩盖方案试图为丢失的分组产生一个与初始分组类似的替代物。正如在 [Perkins 1998] 中讨论的那样，因为音频信号（特别是语音）呈现出大量的短期自相似性，故该方案是可行的。正因为如此，这些技术适合于工作在相对小的丢包率（低于 15%）和小分组（4~40ms）的情况。当丢失长度接近音素的长度（5~100ms）时，这些技术就失效了，因为整个音素可能被听者错过。

也许基于接收方的恢复的最简单方式是分组重复。即用在丢失之前刚到达的分组的副本来代替丢失的分组。这种方法的计算复杂度低，并且工作得相当好。基于接收方恢复的另一种形式是内插法，它使用在丢失之前和之后的音频内插形成一个合适分组来隐藏丢失。内插法比分组重复稍微好一些，但是显然需要更高的计算强度 [Perkins 1998]。

9.3.4 学习案例：使用 Skype 的 VoIP

Skype 是一个非常流行的 VoIP 应用，每天都有超过 5000 万个活跃的账户。Skype 除了提供主机到主机的 VoIP 服务，还提供主机到电话的服务，电话到主机的服务，以及多方主机到主机的视频会议服务。（这里，主机仍然是任一种因特网连接 IP 设备，包括 PC、平板电脑和智能手机。）Skype 于 2011 年被微软公司收购。

因为 Skype 协议是专用的，并且因为所有 Skype 的控制和媒体分组是加密的，所以精确地确定 Skype 的工作过程是非常困难的。无论如何，从 Skype 的 Web 网站和几项测量研究，研究人员已经知道了 Skype 总体上是怎样工作的 [Baset 2006; Guha 2006; Chen 2006; Suh 2006; Ren 2006; Zhang X 2012]。对于语音和视频，Skype 客户都有许多自行支配的不同编解码器，这些编解码器能够以宽泛的速率和质量对媒体进行编码。例如，测量表明 Skype 的视频速率从用于低质量会话的低至 30kbps 到用于高质量会话的高至 1Mbps

左右［Zhang X 2012］。一般而言，Skype语音质量好于由有线电话系统提供的“POTS（简单老式电话服务）”的质量。（Skype编解码器通常以16 000样本/秒或更高速率对语音抽样，这提供比POTS更为丰富的音色，POTS的抽样率为8000/秒。）在默认状态下，Skype通过UDP发送音频和视频分组。然而，控制分组经TCP发送，并且当防火墙阻挡UDP流时，媒体分组也通过TCP发送。Skype对于经UDP发送的语音和视频流使用FEC处理丢包恢复。Skype客户还通过改变视频质量和FEC开销，使它所发送的音频和视频流适应当前的网络情况［Zhang X 2012］。

Skype以一些创新方式使用P2P技术，很好地阐述了P2P是如何应用于除内容分发和文件共享之外的应用中的。如同即时讯息那样，主机到主机因特网电话应用的核心内在地应用了P2P技术，因为用户对（即对等方）彼此实时通信。但是Skype也对两个其他重要功能应用了P2P技术，这两个功能是用户定位和NAT穿越。

如图9-7所示，Skype中的对等方（主机）组织成为一个等级制覆盖网络，其中每个对等方分类为超级对等方和普通对等方。Skype维护一个索引，该索引将Skype用户名映射为当前的IP地址（和端口号）。该索引经过超级对等方分发。当Alice要呼叫Bob时，她的客户搜索该分布式索引以决定Bob的当前IP地址。因为Skype协议是专用的，所以当前并不知道该索引映射是怎样跨越这些超级对等方进行组织的，尽管采用某种形式的DHT组织结构是非常可能的。

P2P技术也被用于Skype**中继**（relay）中，中继对于创建家庭网络中主机之间的呼叫是有用的。许多家庭网络配置提供通过NAT接入因特网，如第4章所讨论的那样。前面讲过NAT防止来自家庭网络外部的主机发起的对家庭网络内部主机的连接。如果两个Skype呼叫方都具有NAT，则存在一个问题，即任一方都不能接受由其他一方发起的呼叫，使得呼叫看起来不可能实现。明智地使用超级对等方和中继很好地解决了这个问题。假设当Alice注册进入系统，她被指派了一个非NAT的超级对等方并对那个超级对等方发起一个会话。（因为是Alice发起了该会话，所以她的NAT允许该会话。）这个会话允许Alice和她的超级对等方交换控制报文。当Bob注册进入系统时发生了同样的事情。此时，当Alice要呼叫Bob，她通知她的超级对等方，超级对等方依次通知Bob的超级对等方，Bob的超级对等方依次通知Bob说“Alice的入呼叫到了”。如果Bob接受了该呼叫，这两个超级对等方选择一个第三方非NAT超级对等方（即中继对等方），中继对等方的工作是中继Alice和Bob的数据。Alice和Bob的超级对等方则分别指示Alice和Bob与该中继发起会话。如图9-7所示，Alice则经过“Alice到中继”连接向该中继发送语音分组（该连接由Alice发起），并且该中继经“中继到Bob”连接转发这些分组（该连接由Bob发起）；从Bob到Alice的分组反方向地流经相同的两条中继连接。瞧！Bob和Alice有了一条端到端连接，即使他们都不能

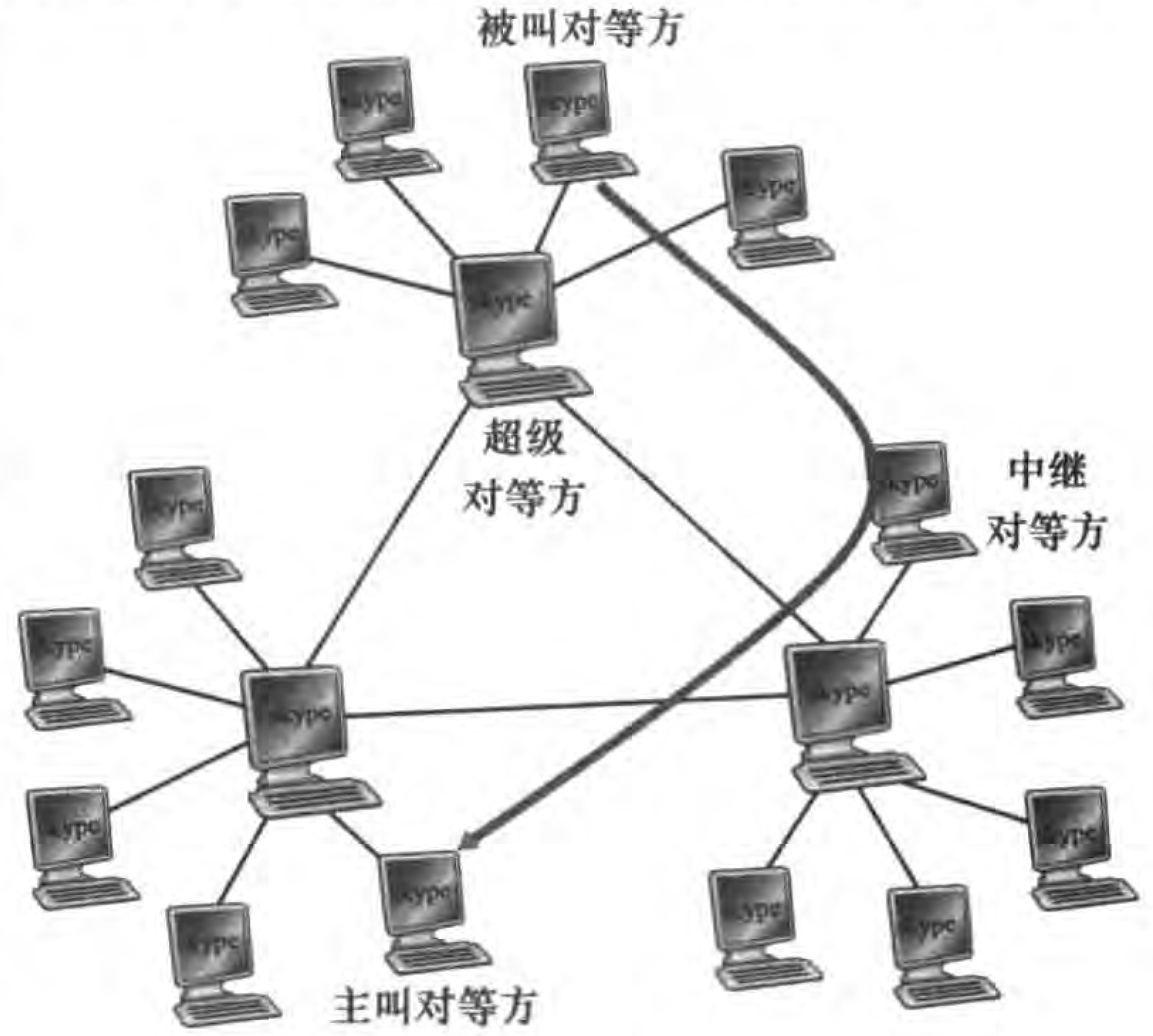


图9-7　Skype对等方

接受一条源于外部的会话。

到现在为止，我们有关 Skype 的讨论关注涉及两人的呼叫。现在我们观察多方音频会议呼叫。对于 $N>2$ 个参与者，如果每个用户希望向每个其他 $N-1$ 个用户发送它的音频流的一个副本，则为支持该音频会议总共 $N(N-1)$ 个音频流将需要发送到网络中去。为了减少这种带宽使用，Skype 应用了一种明智的发送技术。具体而言，每个用户向会议发起方发送它的音频流。会议发起方将这些音频流结合为一个流（基本上是将所有的音频信号加在一起），然后再向每个其他 $N-1$ 个参与者发送每个结合流的一个副本。以这种方式，流的数量被减少到 $2(N-1)$ 条。对普通的两人视频会议，Skype 路由对等方到对等方呼叫，除非需要 NAT 穿越，此时呼叫通过一个非 NAT 对等方中继，如前面所述。对于一个涉及 $N>2$ 个参与者的视频会议呼叫，由于视频媒体的性质，Skype 不像对语音呼叫那样在一个位置将呼叫结合进一条流中，然后将流向所有参与者重新分发。相反，每个参与者的视频流被路由到一个服务器集群（2011 年该服务器集群位于爱沙尼亚），该集群依次将 $N-1$ 个其他参与者的 $N-1$ 条流中继到每个参与者［Zhang X 2012］。你可能想知道为什么每个参与者向服务器而不是向每个其他 $N-1$ 其他参与者直接发送其视频流的副本呢？的确，对两种方法而言，$N(N-1)$ 个视频流正由会议中的 N 个参与者共同接收。其原因是，在大多数接入链路中上行链路带宽比下行链路带宽要低得多，上行链路可能不能支持使用 P2P 方法的 $N-1$ 条流。

诸如 Skype、QQ 和 Google Talk 等 VoIP 系统引发了对隐私性的新关注。具体而言，当 Alice和 Bob 经过 VoIP 通信，Alice 能够嗅探到 Bob 的 IP 地址，进而使用地理定位服务［MaxMind 2016；Quova 2016］来确定 Bob 的当前位置和 ISP（例如他的工作或家庭 ISP）。事实上，Alice 使用 Skype 能够在呼叫创建期间阻挡特定的分组传输，这样她获得 Bob 当前（比如说每个小时）的 IP 地址，而 Bob 并不知道他正被跟踪，并且 Alice 不在 Bob 的接触列表上。此外，从 Skype 发现的 IP 地址能被关联到在 BitTorrent 中发现的 IP 地址，因此 Alice 能够确定 Bob 正在下载的文件［LeBlond 2011］。此外，通过进行流中分组长度的流量分析，可能部分解密一个 Skype 呼叫［White 2011］。

9.4 实时会话式应用的协议

实时会话式应用（包括 VoIP 和视频会议）引人入胜并且非常流行。因此标准机构如 IETF 和 ITU 多年来一直忙于（而且要继续忙下去！）苦心推敲这类应用的标准是毫不奇怪的。借助于实时会话式应用的适当标准，各个独立公司正在创造新的能够互相操作的产品。在本节中，我们探讨用于实时会话式应用的 RTP 和 SIP。这两个标准正广泛地应用于工业产品中。

9.4.1 RTP

在前一节中，我们知道 VoIP 应用的发送端在将块传递给运输层之前为它们附加上首部字段。这些首部字段包括了序号和时间戳。因为大多数多媒体网络应用能够利用序号和时间戳，因此有一个包括音频/视频数据、序号、时间戳以及其他潜在有用字段的标准分组结构是方便的。定义在 RFC 3550 中的 RTP 就是这样一个标准。RTP 能够用于传输通用格式，如用于声音的 PCM、ACC 和 MP3，用于视频的 MPEG 和 H. 263。它也可以用于传输专用的声音和视频格式。目前，RTP 在许多产品和研究原型中得到广泛实现。它也是其他

重要的实时交互协议（如 STP）的补充。

本节我们介绍 RTP。我们也鼓励读者去访问 Henning Schulzrinne 的 RTP 站点 [Schulzrinne-RTP 2012]，该网站提供有关这个主题的很多信息。读者也可以访问 RAT 站点 [RAT 2012]，它记载了使用 RTP 的 VoIP 应用。

1. RTP 基础

RTP 通常运行在 UDP 之上。发送端在 RTP 分组中封装媒体块，然后在 UDP 报文段中封装该分组，然后将该报文段递交给 IP。接收端从 UDP 报文段中提取出这个 RTP 分组，然后从 RTP 分组中提取出媒体块，并将这个块传递给媒体播放器来解码和呈现。

举例来说，考虑使用 RTP 来传输语音。假设语音源采用了 64kbps 的 PCM 编码（也就是采样、量化和数字化）。再假设应用程序在 20ms 块中收集这些编码数据，也就是一个块中有 160 字节。发送端在每个语音数据块的前面加上一个 **RTP 首部**（RTP header），这个首部包括音频编码的类型、序号和时间戳。RTP 首部通常是 12 字节。音频块和 RTP 首部一起形成 **RTP 分组**（RTP packet）。然后向 UDP 套接字接口发送该 RTP 分组。在接收端，应用程序从它的套接字接口收到该 RTP 分组，从 RTP 分组中提取出该音频块，并且使用 RTP 分组的首部字段来适当地解码和播放该音频块。

如果一个应用程序集成了 RTP，而非一种提供负载类型、序号或者时间戳的专用方案，则该应用程序将更容易和其他网络的多媒体应用程序互操作。例如，如果两个不同的公司都开发 VoIP 软件，并且都在它们的产品中集成了 RTP，则希望使用一种 VoIP 产品的用户能够和使用另一种 VoIP 产品的用户进行通信。在 9.4.2 节，我们将看到 RTP 经常和 SIP 一起使用，SIP 是一种因特网电话的重要标准。

应该强调的是，RTP 并不提供任何机制来确保数据的及时交付，或者提供其他服务质量（QoS）保证；它甚至不保证分组的交付，或防止分组的失序交付。RTP 封装的东西确仅为端系统所见。路由器不区分携带 RTP 分组的 IP 数据报和不携带 RTP 分组的 IP 数据报。

RTP 允许为每个源（例如一架照相机或者一个麦克风）分配一个它自己的独立 RTP 分组流。例如，对于在两个参与者之间的一个视频会议，可能打开 4 个 RTP 流，即两个流传输音频（一个方向一个），两个流传输视频（也是一个方向一个）。然而，在编码过程中很多流行的编码技术（包括 MPEG1 和 MPEG2）将音频和视频捆绑在单个流中。当音频和视频与编码器捆绑时，每个方向只产生一个 RTP 流。

RTP 分组并非限用于单播应用，它们也可以经过一对多和多对多的多播树发送。对于一个多对多的多播会话，所有的会话发送方和源通常使用同样的多播组来发送它们的 RTP 流。在一起使用的 RTP 多播流，例如在视频会议应用中从多个发送方发出的音频和视频流，同属于一个 **RTP 会话**（RTP session）。

2. RTP 分组首部字段

如图 9-8 所示，4 个主要的 RTP 分组首部字段是有效载荷类型、序号、时间戳和源标识符字段。

有效载荷类型	序号	时间戳	同步源标识符	其他字段

图 9-8 RTP 首部字段

RTP 分组中的有效载荷类型字段的长度是 7 比特。对于音频流，有效载荷类型字段用于指示所使用的音频编码类型（例如 PCM、适应性增量调制、线性预测编码）。如果发送方在会话过程中决定改变编码，发送方可以通过该有效载荷类型字段来通知接收方这种变化。发送方可能要通过改变该编码来提高语音质量或者减小 RTP 流比特率。表 9-2 列出了当前 RTP 支持的一些音频有效载荷类型。

表 9-2　RTP 支持的一些音频有效载荷类型

有效载荷类型编号	音频格式	采样速率	速率
0	PCMμ 律	8kHz	64kbps
1	1016	8kHz	4.8kbps
3	GSM	8kHz	13kbps
7	LPC	8kHz	2.4kbps
9	G.722	16kHz	48 ~ 64kbps
14	MPEG 音频	90kHz	—
15	G.728	8kHz	16kbps

对于一个视频流，有效载荷类型用于指示视频编码类型（例如运动 JPEG、MPEG1、MPEG2、H.261）。发送方也可以在会话期间动态改变视频编码。表 9-3 列出了当前 RTP 支持的一些视频有效载荷类型。

表 9-3　RTP 支持的一些视频有效载荷类型

有效载荷类型编号	视频格式	有效载荷类型编号	视频格式
26	运动 JPEG	32	MPEG1 视频
31	H.261	33	MPEG2 视频

其他重要的字段如下：

- *序号字段*。序号字段长为 16 比特。每发送一个 RTP 分组则该序号增加 1，而且接收方可以用该序号来检测丢包和恢复分组序列。例如，如果应用的接收方收到的 RTP 分组流在序号 86 和 89 之间存在一个间隙，那么接收方则知道分组 87 和 88 丢失了。那么接收方能够设法来隐藏该丢失数据。
- *时间戳字段*。时间戳字段长 32 比特。它反映了 RTP 数据分组中的第一个字节的采样时刻。如我们在上一节所见，接收方能够使用时间戳来去除网络中引入的分组时延抖动，提供接收方的同步播放。时间戳是从发送方的采样时钟中获得的。举例来说，对于音频的每个采样周期（例如对于 8kHz 的采样时钟每 125μs 为一个周期）时间戳时钟增加 1；如果该音频应用产生由 160 个编码采样组成的块的话，那么当源激活时，对每个 RTP 分组则时间戳增加 160。即使源未激活，该时间戳时钟也将继续以恒定速率增加。
- *同步源标识符*（SSRC）。SSRC 字段长为 32 比特。它标识了 RTP 流的源。通常在 RTP 会话中的每个流都有一个不同的 SSRC。SSRC 不是发送方的 IP 地址，而是当新的流开始时源随机分配的一个数。两个流被分配相同 SSRC 的概率是很小的。如果发生了，这两个源应当选择一个新的 SSRC 值。

9.4.2 SIP

定义在［RFC 3261；RFC 5411］中的**会话发起协议**（Session Initiation Protocol，SIP）

是一个开放和轻型的协议，其功能如下：

- 提供了在主叫者和被叫者之间经 IP 网络创建呼叫的机制。它允许主叫者通知被叫者它要开始一个呼叫。它允许参与者约定媒体编码，也允许参与者结束呼叫。
- 提供了主叫者确定被叫者的当前 IP 地址的机制。因为用户可能动态地分配到地址（使用 DHCP），而且因为它们可能有多个 IP 设备，每个都有一个不同的 IP 地址，所以用户不具有单一的、固定的 IP 地址。
- 提供了用于呼叫管理的机制，这些机制包括在呼叫期间增加新媒体流、在呼叫期间改变编码、在呼叫期间邀请新的参与者、呼叫转移和呼叫保持等。

1. 向已知 IP 地址建立一个呼叫

为了理解 SIP 的要素，最好看一个具体的例子。在这个例子中，Alice 在使用 PC，并且她要呼叫 Bob，Bob 也在使用 PC 工作。Alice 和 Bob 的 PC 都配有基于 SIP 的软件来产生和接收电话呼叫。在这个初始的例子中，我们将假设 Alice 知道 Bob PC 的 IP 地址。图 9-9 描述了这个 SIP 呼叫建立的过程。

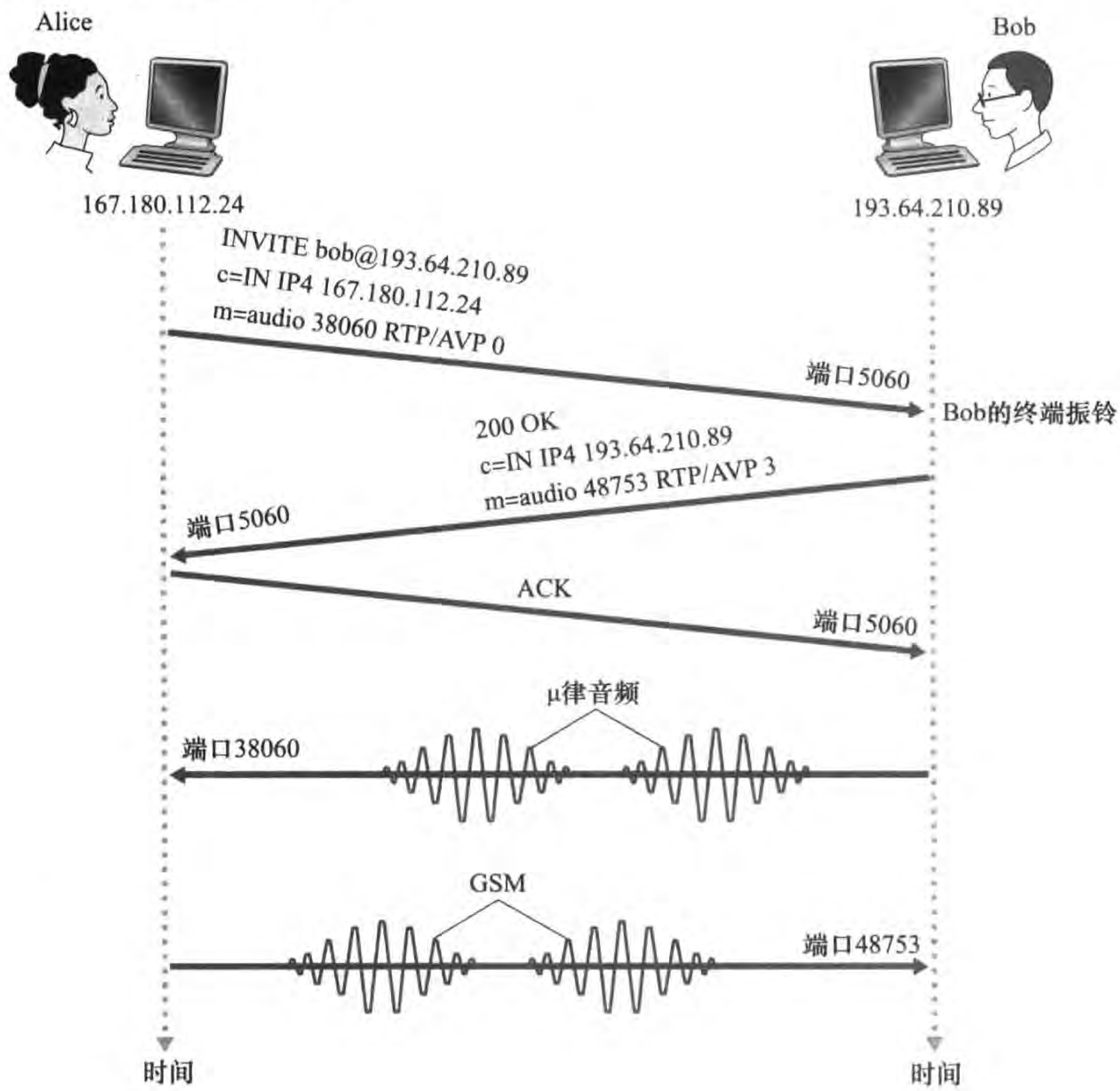


图 9-9 当 Alice 知道 Bob 的 IP 地址时的 SIP 呼叫建立过程

在图 9-9 中，我们看到当 Alice 给 Bob 发送一个 INVITE 报文（这类似于 HTTP 请求报文）时，一个 SIP 会话开始了。该 INVITE 报文通过 UDP 发送给 SIP 的周知端口 5060。（SIP

报文也可以经 TCP 发送。）该 INVITE 报文包括了对 Bob 的标识（bob@ 193.64.210.89）、Alice 当前 IP 地址的指示、Alice 希望接收的音频的指示（该音频以格式 AVP 0 编码，即 μ 律 PCM 编码，并在 RTP 中封装），以及她希望在端口 38060 接收 RTP 分组的指示。在收到了 Alice 的 INVITE 报文之后，Bob 发送一个 SIP 响应报文（该报文类似 HTTP 的响应报文）。该响应 SIP 报文也被发送到 SIP 端口 5060。Bob 的响应包括一个 200 OK 和他的 IP 地址的指示、他希望接收的编码和分组化，以及音频数据应该发送到达的端口号。注意到在这个例子中，Alice 和 Bob 准备使用不同的音频编码机制：要求 Alice 使用 GSM 来对她的音频编码，而要求 Bob 使用 μ 律 PCM 对他的音频编码。在接收到 Bob 的响应后，Alice 向 Bob 发送 SIP 确认报文。在这个 SIP 事务之后，Bob 和 Alice 可以交谈了。（为了看起来清晰，图 9-9 显示 Alice 在 Bob 之后说话，但是事实上他们通常可以同时进行交谈。）Bob 会根据要求对音频进行编码和分组化，并将音频分组发送给 IP 地址 167.180.112.24 的端口号 38060。Alice 也会根据要求对音频进行编码和分组化，并将音频分组发送给 IP 地址 193.64.210.89 的端口号 48753。

从这个简单的例子我们学到了一些 SIP 的关键特性。第一，SIP 是一个带外协议，即发送和接收 SIP 报文使用了一个不同于发送和接收媒体数据的套接字。第二，SIP 报文本身是可读的 ASCII，这与 HTTP 报文类似。第三，SIP 要求所有的报文都要确认，因此它能够在 UDP 或者 TCP 上运行。

在这个例子中，我们考虑一下如果 Bob 没有 μ 律 PCM 编解码器用于音频编码将会发生什么情况。在这种情况下，Bob 不用 200 OK 来响应，而可能用一个 606 Not Acceptable（不可接受）来响应，并在报文中列出他能够使用的所有编解码。然后 Alice 从中选择一个编解码，并发送另一个 INVITE 报文，以此通告了已选择的编解码器。Bob 也能够直接通过发送某个拒绝应答代码来直接拒绝该呼叫。（有很多这种代码，包括“busy（忙）”“gone（离开）”“payment（付费）”和“forbidden（禁止）”。）

2. SIP 地址

在前面的例子中，Bob 的 SIP 地址是 sip：bob@ 193.64.210.89。然而，我们希望许多（即使不是大多数）SIP 地址类似于电子邮件地址。例如，Bob 的地址可以是 sip：bob@ domain.com。当 Alice 的 SIP 设备发送 INVITE 报文，该报文包括这种类似于电子邮件地址的地址；然后 SIP 的基本设施将该报文转发给 Bob 正在使用的 IP 设备（如我们下面要讨论的那样）。其他可能的 SIP 地址形式可以是 Bob 过去的电话号码或者只是 Bob 的名字/中间名/姓氏（假设它是唯一的）。

SIP 地址的一个有趣特点是它们能够被包括在 Web 页面中，就像人们的电子邮件地址用 mailto URL 形式包含在 Web 页面中那样。例如，假设 Bob 有个人主页，并且他要为这个主页的访问者提供一个呼叫他的方法。于是他可能只是在主页中包括该 URL sip：bob@ domain.com。当访问者点击该 URL，访问者设备中的 SIP 应用将启动，并向 Bob 发送 INVITE 报文。

3. SIP 报文

在这个 SIP 简短的介绍中，我们无法包括所有 SIP 报文类型和首部，而只是简要地浏览一下 SIP INVITE 报文，以及少数通用的首部行。我们再假设 Alice 要对 Bob 发起 VoIP 呼叫，此时 Alice 只知道 Bob 的 SIP 地址 bob@ domain.com，并不知道 Bob 正在使用的设备的 IP 地址。那么她的报文可能看起来有些像下面这个：

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

这个 INVITE 行包括 SIP 的版本，这与 HTTP 请求报文一样。任何时候 SIP 报文通过一个 SIP 设备（包括产生该报文的设备）时，它附加上一个 Via 首部来指示该设备的 IP 地址。（我们不久将看到通常 INVITE 报文在到达被叫者的 SIP 应用之前会通过很多 SIP 设备。）与电子邮件报文类似，SIP 报文包括一个 From 首部行和一个 To 首部行。该报文包括一个 Call-ID（呼叫标识符），它唯一地标识该呼叫（类似电子邮件中的报文 ID）；包括一个 Content-Type（内容类型）首部行，定义用于描述包含在 SIP 报文中的内容的格式；还包括 Content-Length（内容长度）首部行，提供报文中内容的字节长度。最后，在一个回车和换行之后，该报文包含内容部分。在这个例子中，内容提供了有关 Alice 的 IP 地址和 Alice 要如何接收该音频的信息。

4. 名字翻译和用户定位

在图 9-9 的例子中，我们假设 Alice 的 SIP 设备知道能够联系到 Bob 的 IP 地址。但是这种假设是很不真实的，不仅因为 IP 地址通常是通过 DHCP 动态分配的，而且 Bob 可能有多个 IP 设备（例如，在家里、工作中和汽车里有不同的设备）。因此现在我们假设 Alice 只知道 Bob 的电子邮件地址 bob@domain.com，而且对基于 SIP 的呼叫使用同样的地址。在这种情况下，Alice 需要获得用户 bob@domain.com 正在使用的设备的 IP 地址。为了获得它，Alice 创建一个 INVITE 报文，它以 INVITE bob@domain.com SIP/2.0 开始，并将该报文发送给一个 **SIP 代理**（SIP proxy）。该代理将以一个 SIP 回答来响应，该回答中也可能包含 bob@domain.com 正在使用的设备的 IP 地址。该回答也可以选择包括 Bob 的语音信箱的 IP 地址，或者可能包括一个 Web 页面的 URL（上面写着“Bob 在睡觉，请不要打扰！”）。代理返回的结果也可能取决于呼叫者：如果呼叫是 Bob 的妻子发出的，它可能接受该呼叫，并提供 IP 地址；如果呼叫是 Bob 的岳母发出的，它可能用指向“我正在睡觉”的 Web 页面的 URL 来响应。

现在，你可能想知道这个代理服务器是怎样确定 bob@domain.com 现在的 IP 地址的？为了回答该问题，我们需要讲一下另一个 SIP 设备，即 **SIP 注册器**（SIP registrar）。每个 SIP 用户都有一个相关联的注册器。任何时候用户在设备上发起 SIP 应用时，该应用给注册器发送一个 SIP 注册报文，通知注册器它现在的 IP 地址。例如，当 Bob 在他的 PDA 上发起 SIP 应用时，该应用将发送一个类似于下述内容的报文：

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

Bob 的注册器跟踪 Bob 现在的 IP 地址。无论何时 Bob 切换到一个新的 SIP 设备时，该新设备将发送一个新的注册报文，指示该新的 IP 地址。如果 Bob 长时间使用同样的设备，该设备将发送刷新注册报文，指示最近发送的 IP 地址仍然有效。（在上面的例子中，需要

每隔 3600 秒发送刷新报文来维护在注册器中的地址。）值得注意的是注册器和 DNS 权威名字服务器类似：DNS 服务器将固定的用户名翻译到固定的 IP 地址；SIP 注册器把固定的人识别标志（例如 bob@ domain. com）翻译为一个动态的 IP 地址。SIP 注册器和 SIP 代理通常运行在同一台主机上。

现在我们检查一下 Alice 的 SIP 代理服务器是如何获得 Bob 当前的 IP 地址的。从上面的讨论我们看到，代理服务器只需要转发 Alice 的 INVITE 报文给 Bob 的注册器/代理即可。然后注册器/代理将该报文转发给 Bob 现在的 SIP 设备。最后，在已收到了 Alice 的 INVITE 报文之后，Bob 可以向 Alice 发送一个 SIP 应答。

举例来说，考虑图 9-10，其中正在 217. 123. 56. 89 上工作的 jim@ umass. edu 要对正在 197. 87. 54. 21 上工作的 keith@ upenn. edu 发起 IP 语音（VoIP）会话。采用下面的步骤：①Jim 向 umass 的 SIP 代理发送 INVITE 报文。②该代理在 SIP 注册器 upenn. edu 上进行 DNS 查询（在图中没有显示），并将该报文转发给注册服务器。③因为 keith@ upenn. edu 没有在注册器 upenn 上注册，upenn 注册器发送一个重定向应答，指示它应该试一试 keith@ nyu. edu。④umass 的代理向 NYU 的 SIP 注册器发送 INVITE 报文。⑤NYU 注册器知道 keith @ upenn. edu 的 IP 地址，并将 INVITE 报文转发给正运行 Keith 的 SIP 客户的主机 197. 87. 54. 21。⑥～⑧通过注册器/代理把 SIP 响应返回给 217. 123. 56. 89 上的 SIP 客户。⑨媒体直接在两个客户之间发送。（还有一个 SIP 确认报文，图中没有画出来。）

我们有关 SIP 的讨论集中在语音呼叫的呼叫发起方面。SIP 通常是一个发起和结束呼叫的信令协议，它能够用于视频会议呼叫和基于文本的会话。事实上，SIP 已经成为许多即时讯息应用的基本组件。我们鼓励希望学习更多有关 SIP 知识的读者访问 Henning Schulzrinne 的 SIP Web 站点［Schulzrinne-SIP 2016］。特别是，你在这个站点上会发现 SIP 客户和服务器的开源软件［SIP Software 2016］。

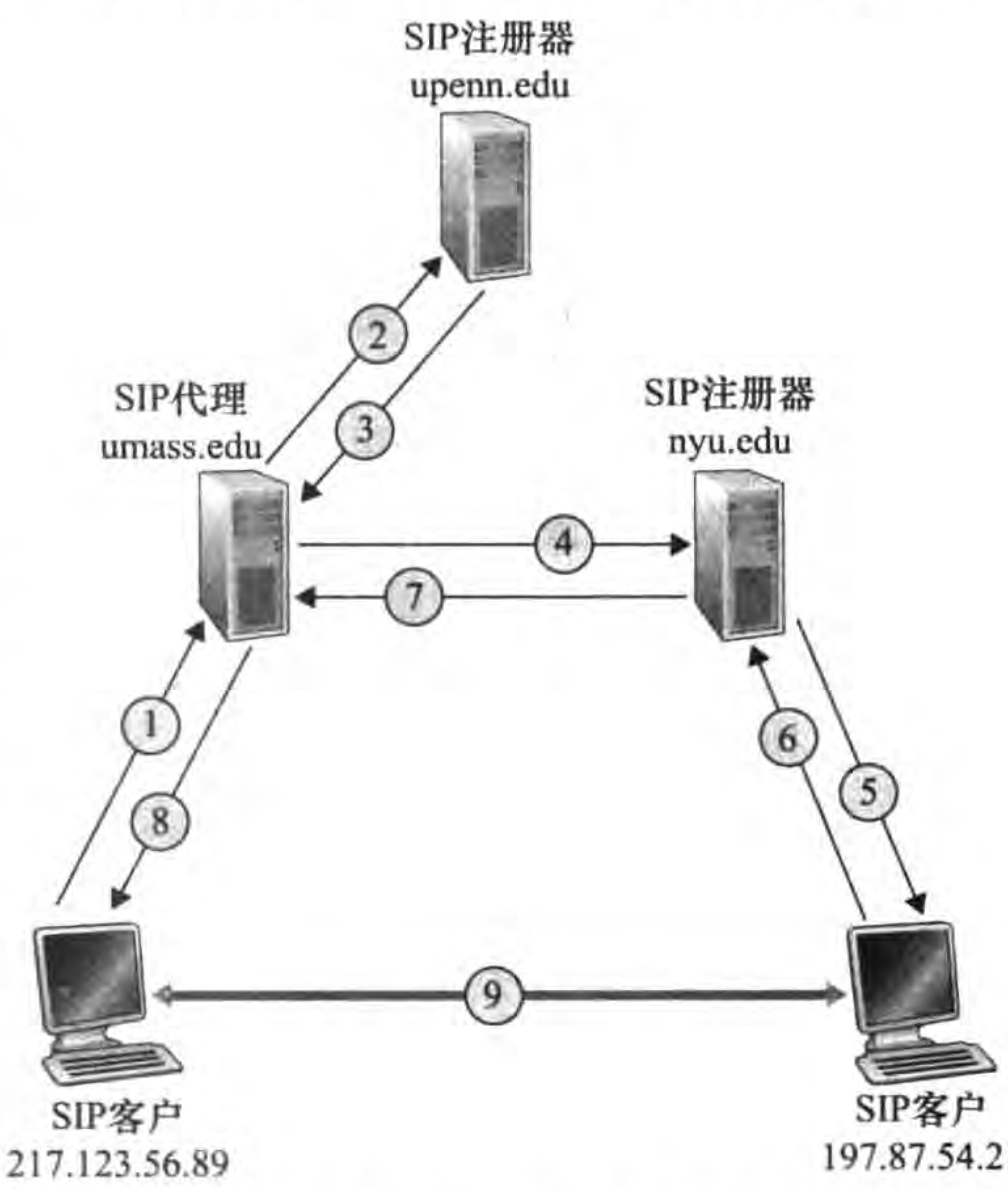


图 9-10　涉及 SIP 代理与注册器的会话发起

9.5　支持多媒体的网络

在 9.2～9.4 节中，我们学习了诸如客户缓存、预取、对可用带宽的适应性媒体质量、适应性播放和丢包缓解技术等应用级机制如何用于多媒体应用，以改善多媒体应用的性能。我们也学习了内容分发网和 P2P 覆盖网络如何用于提供系统级的交付多媒体内容的方法。这些技术和方法都被设计用于今天的尽力而为因特网。的确，现在因特网得到应用就是因为它只提供单一的、尽力而为类型的服务。但是作为计算机网络的设计者，我们禁不住要问：网络（而不是应用程序或者仅应用级的基础设施）是否可以提供支持多媒体内容交付的机制。如我们很快将看到的那样，答案当然是肯定的！但是我们也将看到，许多这

样的新网络级机制还没有得到广泛部署。这可能是由于它们的复杂性和下列事实所致：应用级基础设施以及尽力而为服务和适当定制的网络资源（例如带宽）的确能够提供“足够好的”（即使不总是尽善尽美的）端到端多媒体交付服务。

表9-4总结了能够对多媒体应用提供网络层支持的三种宽泛的方法。

表9-4　支持多媒体应用的三种网络层方法

方法	粒度	保证	机制	复杂性	当前部署
尽可能利用尽力而为服务	公平处理所有流量	无或者软	应用级支持，CDN，覆盖网络，网络级资源供给	最小	无处不在
区分服务	不同类型的流量处理不同	无或者软	分组标识，监管，调度	中等	某些
每连接服务质量（QoS）保证	每个源到目的地流处理不同	一旦流被准入，软或者硬	分组标识，监管，调度，呼叫准入和信令	高	很少

- *尽可能利用尽力而为服务*。我们在9.2~9.4节学习的应用级机制和基础设施能够成功地用于定制良好的网络——该网络中偶然出现丢包和过大的端到端时延。当预见到需求增加，ISP部署额外的带宽和交换能力以持续确保满意的时延和丢包性能［Huang 2005］。我们将在9.5.1节中进一步讨论**网络定制**（network dimensioning）。
- *区分服务*。自因特网的早期，就已经设想不同类型的流量（例如，在IPv4分组首部中服务类型字段所指示的类型）能够由不同类型的服务所提供，而不是单一的“以不变应万变”的尽力而为服务。使用**区分服务**（differentiated service），当两类流量在一台路由器中排队时，一种类型的流量可以给定严格的优于另一种类型的流量的优先权。例如，属于实时会话式应用的分组由于其严格的时延限制，可能会给定优于其他分组的优先权。在网络中引入区分服务将要求一些用于分组标记（指示一个分组的服务类型）、分组调度和其他方面的新机制。我们将在9.5.2节和9.5.3节中涉及区分服务以及实现这种服务的新网络机制。
- *每连接服务质量（QoS）保证*。使用每连接QoS保证，每个应用的实例显式地预约端到端带宽，并因此具有确保的端到端性能。**硬保证**（hard guarantee）意味着应用将必定接收到它所请求的服务质量。**软保证**（soft guarantee）意味着应用将以高概率接收到它所请求的服务质量。例如，如果某用户要从主机A向主机B进行VoIP呼叫，该用户的VoIP应用在两台主机之间沿着路径在每条链路上显式地预留带宽。但是，允许应用做预约和请求网络同意该预约，这需要一些大的变化。首先，我们需要一个协议来代表应用程序，从发送方到其接收方沿路径预约链路带宽。第二，在路由器队列中将需要新的调度策略，使每连接带宽预约能够兑现。最后，为了进行预约，应用程序必须向网络给出描述来说明它们希望发送进网络的流量，并且网络将需要监管每个应用程序的流量以确保它遵守这个描述。当这些机制结合时，在主机和路由器中要求新的和复杂的软件。因为每连接QoS保证服务尚未见到大规模部署，我们将仅在9.5.4节中简要地涉及这些机制。

9.5.1　定制尽力而为网络

从根本上说，支持多媒体应用的困难是由其严格的性能要求引起的，即低的端到端分

组时延、时延抖动和丢包，而事实是，无论何时网络变得拥塞，都将出现较大的分组时延、时延抖动和丢包。改善多媒体应用质量的第一种方法就是“在问题上砸钱”，因此直接避免资源竞争即可。这种方法常用于解决有关资源受限的任何问题。在网络多媒体的场合，这意味着在整个网络中提供充足的链路带宽，使网络拥塞及其导致的分组时延和丢失决不会（或仅非常少地）出现。具有充足的链路带宽，分组将很快地通过今天的因特网而没有排队时延或丢失。从许多方面看，这是一种理想的情况：完美地执行多媒体应用，用户是幸运的，所有要求都能够满足而不改变因特网尽力而为的体系结构。

当然，问题是为实现这种极乐世界提供多大容量才是“充足的”，以及提供“充足的”带宽的成本从 ISP 的商业角度来说是否实际。在一个给定拓扑中为网络链路提供多大容量以取得给定水平的性能的问题常被称为**带宽供给**（bandwidth provisioning）。如何设计一个网络拓扑（其中放置一些路由器，如何用链路互联这些路由器，并为链路分配容量）以取得给定水平的端到端性能这个更为复杂的问题常被称为**网络定制**（network dimensioning）。带宽供给和网络定制都是复杂的专题，它们远超过了本教科书的范围。然而，我们这里注意到，为了预测两个网络端点之间的应用级性能，必须处理下列问题，并因此提供充足的容量来满足应用的性能要求。

- *网络端点之间的流量需求模型。*这些模型可能需要定义在呼叫层次（例如，用户“到达”网络并启动端到端应用）和分组层次（例如，由进行中的应用所产生的分组）。注意负载可能随着时间而变化。
- *定义良好的性能要求。*例如，为支持诸如会话式多媒体应用等时延敏感的流量，其性能要求可能是：分组的端到端时延大于最大可容忍时延的概率要小于某个很小的值 [Fraleigh 2003]。
- *对给定的负载模型预测端到端性能的模型，以及求出最小成本带宽分配（该带宽分配将导致满足所有用户的需求）的技术。*这里，研究人员正忙于研发能够量化给定负载下的性能的性能模型，以及能求出满足性能要求的最小成本带宽分配的优化技术。

假定今天尽力而为的因特网能够（从技术角度讲）以适当的性能水平支持多媒体流量（如果它被定制成这样的话），自然的问题是为什么今天的因特网满足不了这样的要求。答案基本上是经济上和组织上的原因。从经济角度看，用户将愿意向其 ISP 支付足够多的费用，使 ISP 安装充足的带宽经尽力而为的因特网来支持多媒体应用吗？组织问题也许更为令人气馁。注意到在两个多媒体端点之间的端到端路径将通过多个 ISP 的网络。从组织角度看，这些 ISP 将愿意合作（也许以收入共享方式）以确保端到端路径被适当地定制来支持多媒体应用吗？对这些经济和组织问题的展望，参见 [Davies 2005]。对于供给第一层主干网络以支持时延敏感流量的展望，参见 [Fraleigh 2003]。

9.5.2 提供多种类型的服务

也许对今天因特网中的以不变应万变的尽力而为服务而言，一种最简单的强化是将流量划分为多种类型，并为这些不同类型的流量提供不同等级的服务。例如，某 ISP 可能要为时延敏感的 VoIP 或电信会议流量比为电子邮件或 HTTP 等弹性流量提供更高的服务类型（并对该服务收取更高的费用）。另一种做法是，ISP 可能直接向愿意对这种改进服务支付更多费用的顾客提供更高质量的服务。一些住宅有线接入 ISP 和蜂窝无线接入 ISP 已经采用了这样的梯次等级服务，即铂金卡服务用户比金卡服务用户或银卡服务用户享有更

好的服务性能。

我们都从日常生活中熟悉了不同类型的服务，如航班上头等舱乘客比公务舱乘客得到更好的服务，公务舱乘客又比经济舱乘客得到更好的服务；VIP 在活动中能够立即进入，而所有其他人都必须排队等待；在某些国家中老年人被尊重，提供了荣誉座位和最精细的食物。重要的是注意到在聚合流量中（即在多种流量类型中而不是单个连接中）提供了这种有差别的服务。例如，所有头等舱乘客被一视同仁（没有哪个头等舱乘客得到了比其他头等舱乘客更好的服务），就像所有的 VoIP 分组在网络中得到了相同的对待，与它们所属的特定的端到端连接无关。如我们将见到的那样，通过处理少量的流量聚合，而不是大量的单个连接，提供好于尽力而为服务所需要的新型网络机制能够保持相对简单。

早期因特网设计者的心中清晰地具有这种多种类型服务的概念。回想第 4 章中 IPv4 首部中的服务类型（ToS）字段。IEN123［ISI 1979］描述也呈现在 IPv4 数据报的原型中的 ToS 字段时说："服务类型［字段］提供了所希望的服务质量的抽象参数的指示。当传输一个数据报通过某特定网络时，这些参数被用于引导实际服务参数的选择。几种网络提供了服务优先权，该优先权以某种方式把高优先权流量看得比其他流量更为重要。"在 40 多年前，向不同等级的流量提供不同等级的服务的设想就是清晰的！然而，我们又花费了等长的时间去实现这种愿景。

1. 促进思考的场景

下面用几种促进思考的场景来开始我们的提供多种类型服务的网络机制的讨论。

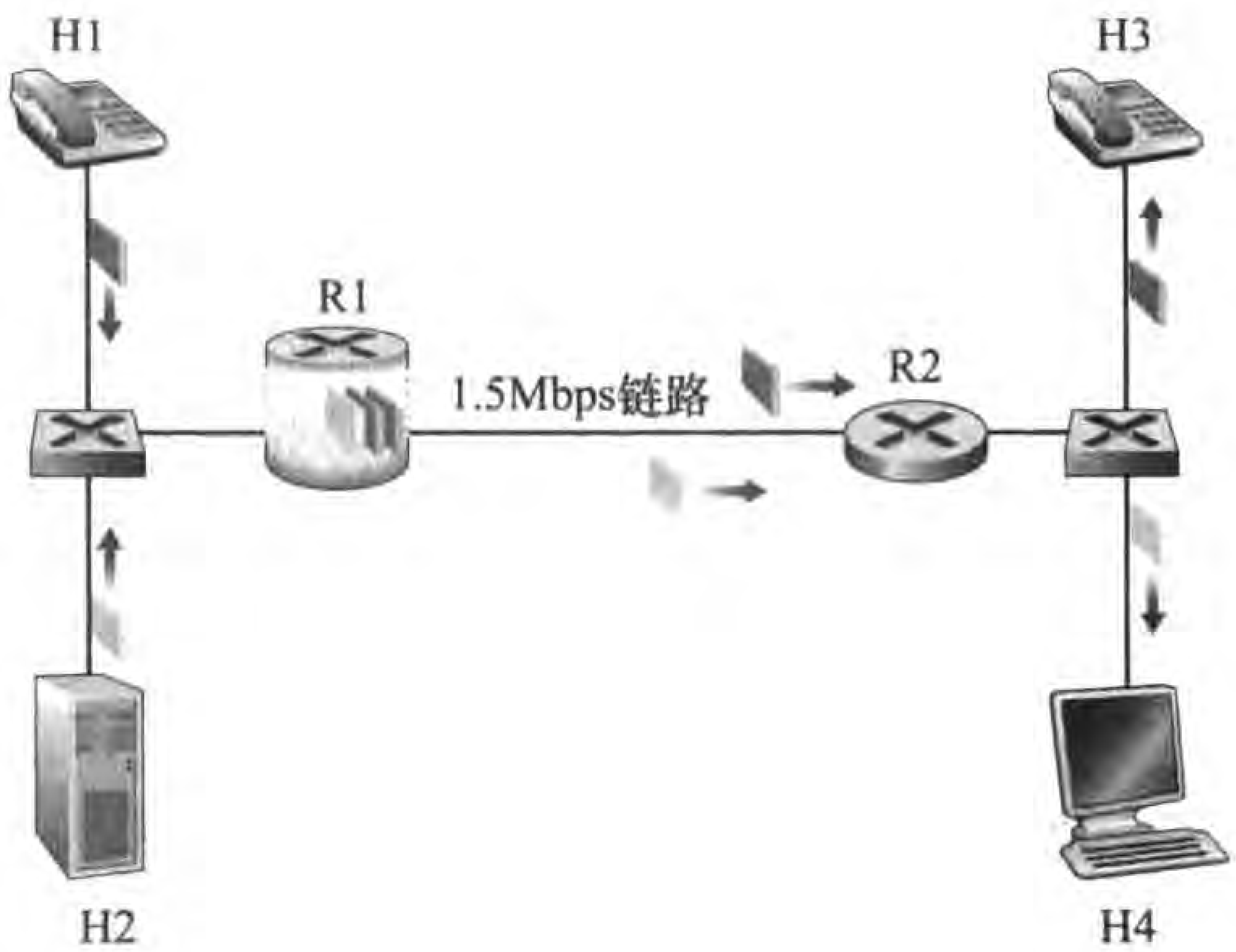


图 9-11　音频应用与 FTP 应用的竞争

图 9-11 表示了一种简单的网络场景，两个应用分组流产生于位于一个局域网的主机 H1 和 H2，它们的目的地是另一个局域网的主机 H3 和 H4。在这两个局域网上的两台路由器通过一条 1.5Mbps 的链路连接起来。我们假设局域网的速度远远高于 1.5Mbps，并且关注路由器 R1 的输出队列；注意到如果 H1 和 H2 的总计发送速率超过了 1.5Mbps，分组时延和丢包将会出现。我们进一步假设 1Mbps 的音频应用（例如一个 CD 质量的音频呼叫）共享 R1 和 R2 之间 1.5Mbps 的链路，同时从 H2 到 H4 有一个 HTTP Web 浏览应用正在下载一个 Web 网页。

在尽力而为服务的因特网中，该音频和 HTTP 分组在 R1 的输出队列中混合，并且（通常）以先进先出（FIFO）的次序传输。在这种情况下，来自 Web 服务器的突发分组可能潜在地填充满这个队列，引起 IP 音频分组过度延迟或者由于 R1 的缓存溢出而丢失。我们应该如何解决这个潜在的问题呢？假定该 HTTP Web 浏览应用没有时间限制，我们的直觉也许是在 R1 为音频分组分配严格的优先级。在一个严格的优先级调度规则下，在 R1 输出缓存的音频分组总是在 R1 输出缓存中的任何 HTTP 分组之前传输。对音频流量而言，从 R1 到 R2 的链路看起来像一条 1.5Mbps 专用链路，而对于 HTTP 流量仅当没有音频流量排队时，才使用 R1 到 R2 的链路。为了让 R1 在它的队列中区分音频和 FTP 分组，每个分组必须被标记为属于这两类流量中的哪一类。这是 IPv4 中服务类型（ToS）字段的最初目的。显而易见，这则是我们对需要提供多种类型流量机制的第 1 个见解。

见解1：**标记分组**（packet marking）使得路由器区分属于不同类型流量的分组。

注意到尽管我们的例子考虑到竞争性的多媒体和弹性流，但是相同的见解应用于实现了铂金卡、金卡和银卡服务类型的情况，即仍需要标记分组机制以指示分组属于哪个类型的服务。

现在假设路由器被配置为给标记为属于1Mbps音频应用的分组赋予高优先级。因为输出链路速度是1.5Mbps，即使HTTP分组得到较低的优先级，它们仍然可以收到平均0.5Mbps的传输服务。但是如果音频应用开始以1.5Mbps或者更高的速率（或者恶意的，或者由于应用的差错）发送分组，那会出现什么样的情况呢？在这种情况下，HTTP分组将挨饿，也就是在R1到R2的链路上得不到任何服务。如果多个应用（例如，多个音频呼叫）都具有同等的服务类型，共享一段链路带宽，那么也会出现类似问题，即它们也可能共同饿死该HTTP会话。理想情况下，一种服务要与各类流量有隔离度，以保护一种流量类型免受其他流量类型干扰。这种保护能够在网络中的不同地方实现，在每台路由器中，在进入网络的首个入口，或在网络边界域间。这则是我们的第2个见解。

见解2：希望在流量类型之间提供**流量隔离**（traffic isolation）的度，以便一类流量不会受到另一类异常流量的负面影响。

我们将考察在流量类型之间提供这种隔离的特定机制。这里我们注意到，有两大类方法可以使用。首先，可以执行如图9-12所示的**流量监管**（traffic policing）方法。如果流量类型或流必须满足一定的准则（例如，音频流不超过1Mbps的峰值速率），那么可以设置一个监管机制来确保这些准则的确被遵守。如果被监管的应用行为异常，这个监管机制将采取某种行动（例如，丢弃或者延时那些违反这些准则的分组），以便实际进入网络的流量符合这些准则。我们很快将考察的漏桶机制也许是使用最广泛的监管机制。在图9-12中，分组分类和标记机制（见解1）以及监管机制（见解2）都一起在网络的边缘实现，或在端系统中实现，或在边界路由器中实现。

为流量类型之间提供隔离的一种互补的方法是，链路级的分组调度机制为每种类型明确地分配固定量的链路带宽。例如，在R1能够给音频类型分配1Mbps，能够给HTTP流分配0.5Mbps。在这种情况下，音频和HTTP流分别看到了容量为1.0Mbps和0.5Mbps的逻辑链路，如图9-13所示。通过严格执行链路级的带宽分配，一种类型

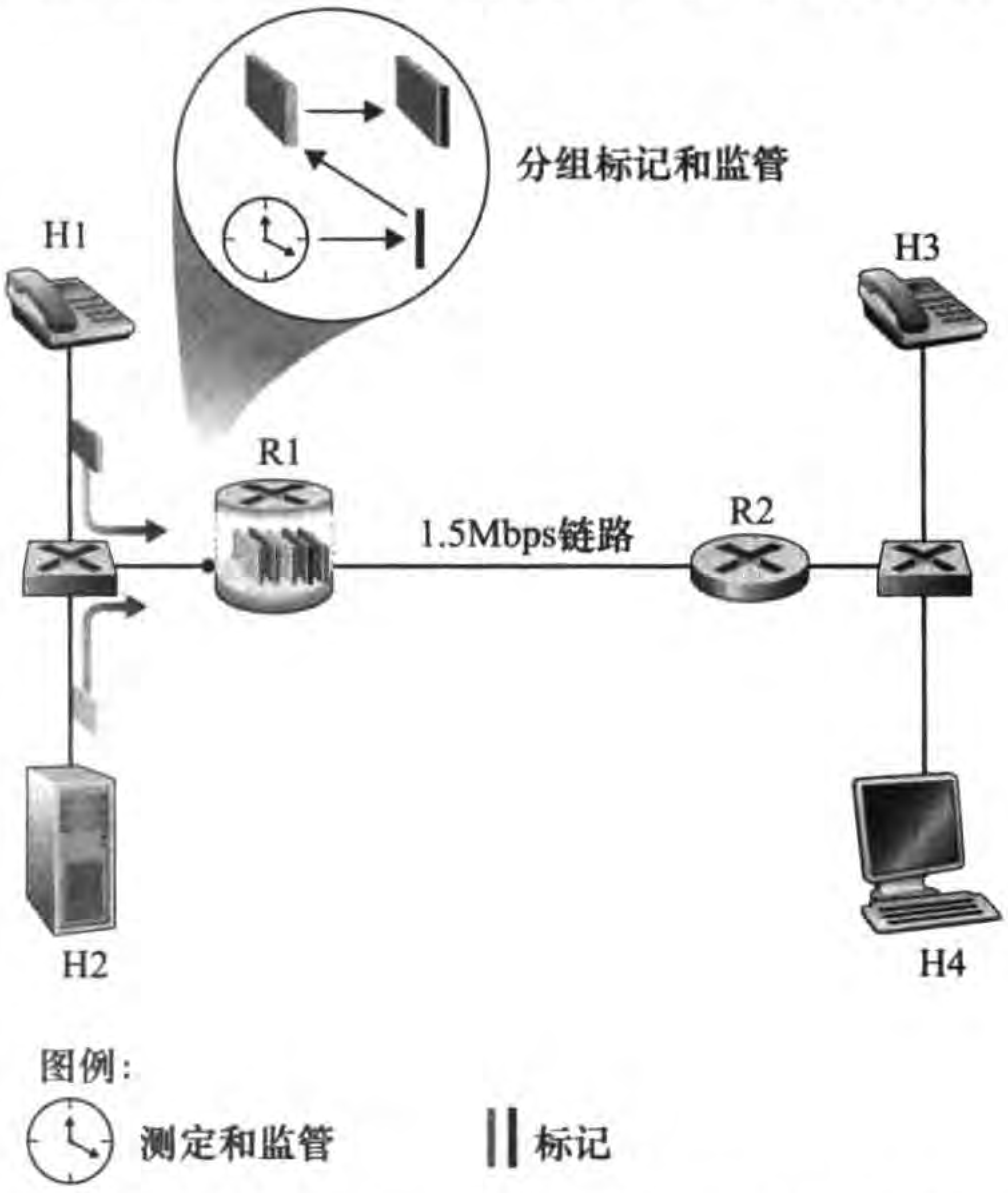


图9-12 监管（与标记）音频与HTTP流量类型

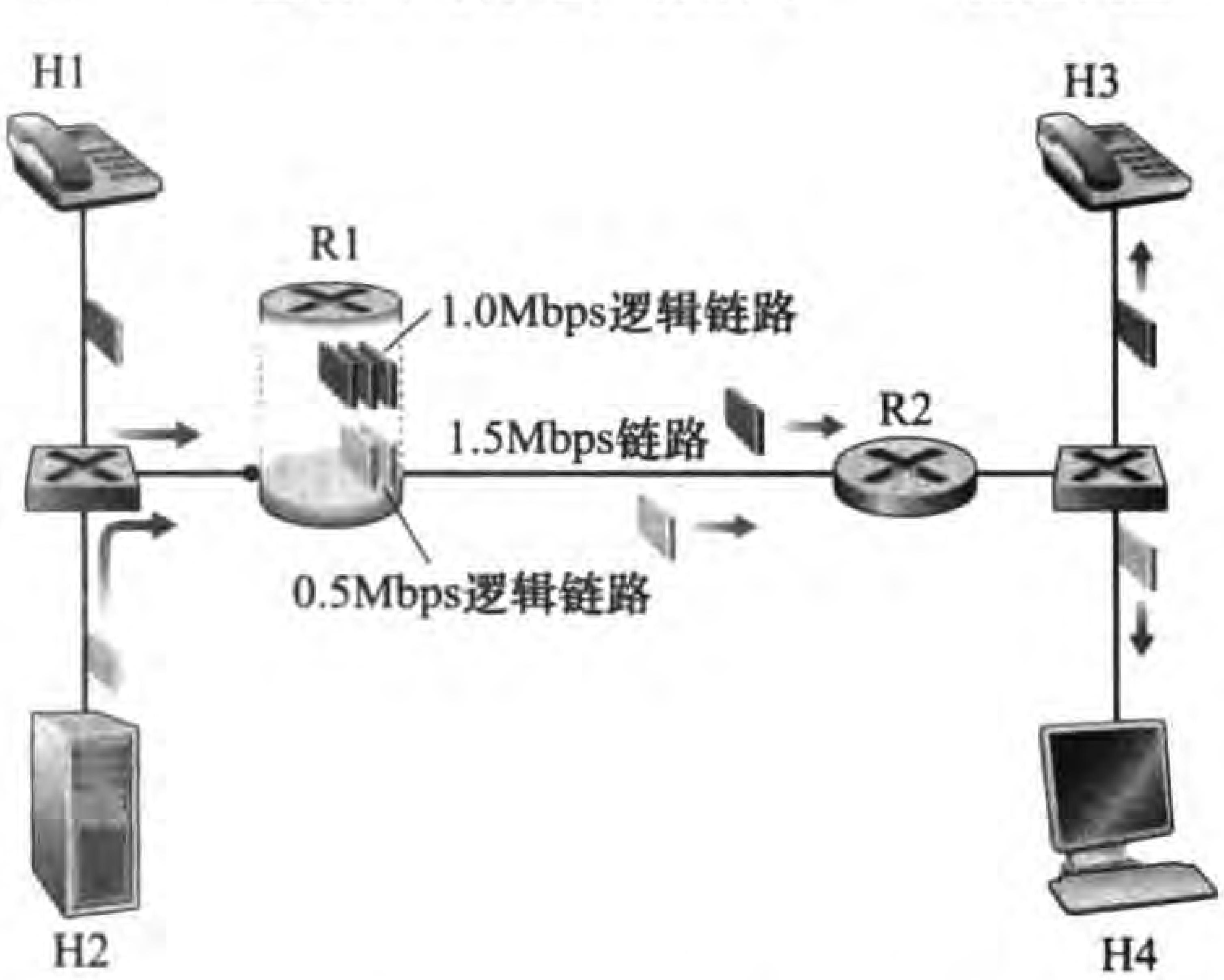


图9-13 音频与HTTP流量类型的逻辑隔离

仅能够使用已经分配的带宽量；特别是，它不能利用其他应用现在未使用的带宽。例如，如果音频流静默了（例如，如果谈话者停顿，不产生音频分组），HTTP 流在 R1 到 R2 的链路上的传输带宽仍然不能够超过 0.5Mbps，即使音频流分配的 1Mbps 带宽在那个瞬间没有使用。由于带宽是一种“使用它或丢失它（use-it-or-lose-it）”的资源，没有理由妨碍 HTTP 流量使用没有由音频流使用的带宽。我们将希望尽可能高效地使用带宽，当能够以别的方法使用它时决不浪费带宽。这引发我们的第 3 个见解。

见解3：当为流量类型或流之间提供隔离时，希望尽可能有效地使用资源（例如链路带宽和缓存）。

2. 调度机制

在 1.3 节和 4.2 节我们曾经讨论过，属于各种网络流的分组被复用在一起，并且在与链路关联的输出缓存排队等待传输。选择在链路上传输的排队分组的方式称为**链路调度规则**（link-scheduling discipline），并在 4.2 节进行了详细讨论。回想在 4.2 节中讨论过三种链路调度规则，即 FIFO、优先权排队和加权公平队列（WFQ）。我们马上将看到 WFQ 对于格力流量类型将起到特别重要的作用。

3. 监管：漏桶

我们前面的见解之一是监管，速率的调节是一种重要的 QoS 机制，以该速率允许一类流量或流（在下面的讨论中，我们将假定监管的单位是流）向网络注入分组。但是应该对一个流的分组速率的哪些方面进行监管呢？我们能够指出 3 个重要的监管准则，每个准则根据被监管分组流的时间范围而互不相同：

- 平均速率。网络可能希望限制一个流的分组能够发送到网络中的长期平均速率（每个时间间隔的分组数）。这里一个关键的问题是监管平均速率的时间间隔。一个平均速率被限制为每秒 100 个分组的流要比一个每分钟 6000 个分组的源受到的约束更严格，即使在一个足够长的时间间隔上它们有相同的平均速率。例如，后者的限制允许一个流在给定 1 秒长的时间间隔内发送 1000 个分组，而前者的限制不允许这种发送行为。
- 峰值速率。平均速率约束限制了一个相对长的时间内能够发送到网络中的流量，而峰值约束限制了一个较短时间内能够发送的最大分组数。使用我们上面的例子，网络能以每分钟 6000 个分组的平均速率来监管一个流，但是限制该流的峰值速率为每秒 1500 个分组。
- 突发长度。网络也许还希望限制极短的时间间隔内能够发送到网络中的最大分组数（分组的“突发（burst）”）。在这个限制中，因为时间间隔长度趋近于0，该突发长度限制了能够瞬间发送到网络中的分组数量。尽管瞬间发送多个分组到网络中在物理上是不可能的（毕竟每条链路都有一个无法超越的物理传输速率！），但对最大突发长度的抽象也是有用的。

漏桶机制是一个能够用来表征这些监管限制的抽象。如图 9-14 所示，漏桶由一个能够容纳 b 个令牌的桶组成。令牌加进该桶的过程如下。可能潜在地加入桶中的新令牌总是以每秒 r 个令牌的速率产生。（为了简单化起见，我们这里假设时间单元是秒。）当产生一个令牌时，如果桶填充得少于 b 个令牌，新产生的令牌加入到该桶中；否则忽略该新产生的令牌，令牌桶保持具有 b 个令牌的满状态。

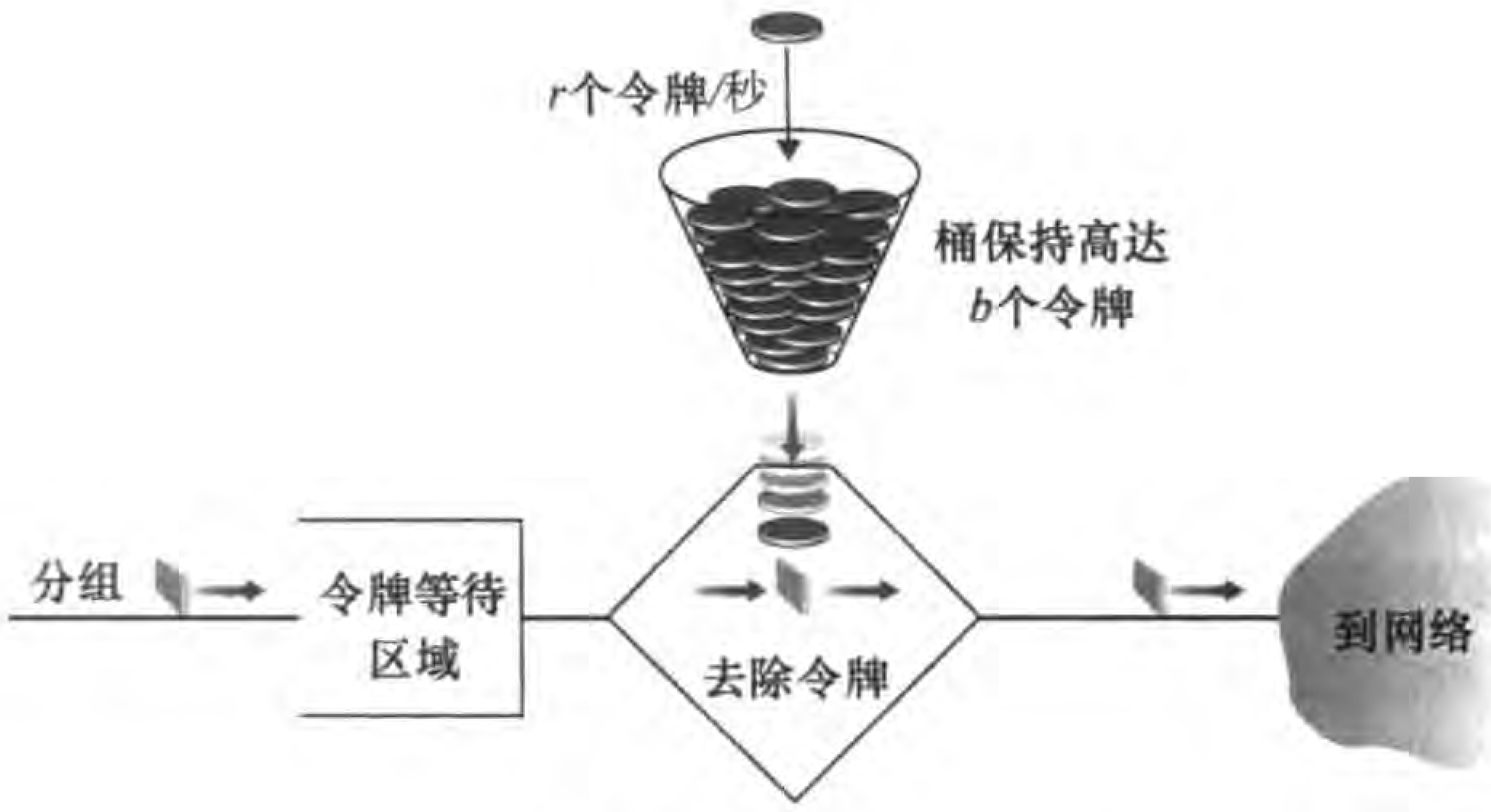


图 9-14 漏桶监管器

现在我们考虑如何用漏桶来监管分组流。假设在一个分组向网络传输之前，必须首先从令牌桶中去除一个令牌。如果令牌桶是空的，分组必须等待一个令牌。（另一种方法是丢弃该分组，尽管我们这里不讨论这种选择。）现在我们考虑这种行为是如何监管一个流量流的。因为在桶中最多能有 b 个令牌，所以漏桶监管的流的最大突发长度是 b 个分组。此外，因为令牌产生速率是 r，所以在任何长度为 t 的时间间隔内能够进入网络中的最大分组数目为 $rt+b$。因此，令牌产生速率 r 用于限制分组能够进入网络的长期平均速率。除了监管长期平均速率之外，使用漏桶（具体来说，串联的两个漏桶）来监管流的峰值速率也是有可能的；见本章末的习题。

4. 漏桶 + 加权公平排队 = 队列中可证明的最大时延

我们关于策略的最后讨论是说明如何将调度和监管两种机制结合起来，以提供通过路由器队列的时延界限（鼓励已经忘记 WFQ 的读者复习一下 4.2 节）。我们考虑一台使用了 WFQ 调度的路由器，它的输出链路多路复用了 n 条流，其中每条流被一个参数为 b_i 和 r_i 的漏桶监管，$i=1, \cdots, n$。这里我们使用的术语“流”大致是指不能被调度器相互区别的分组集合。实践中，一条流可能是由单个端到端连接上的流量或者许多这种连接的集合流量组成的，参见图 9-15。

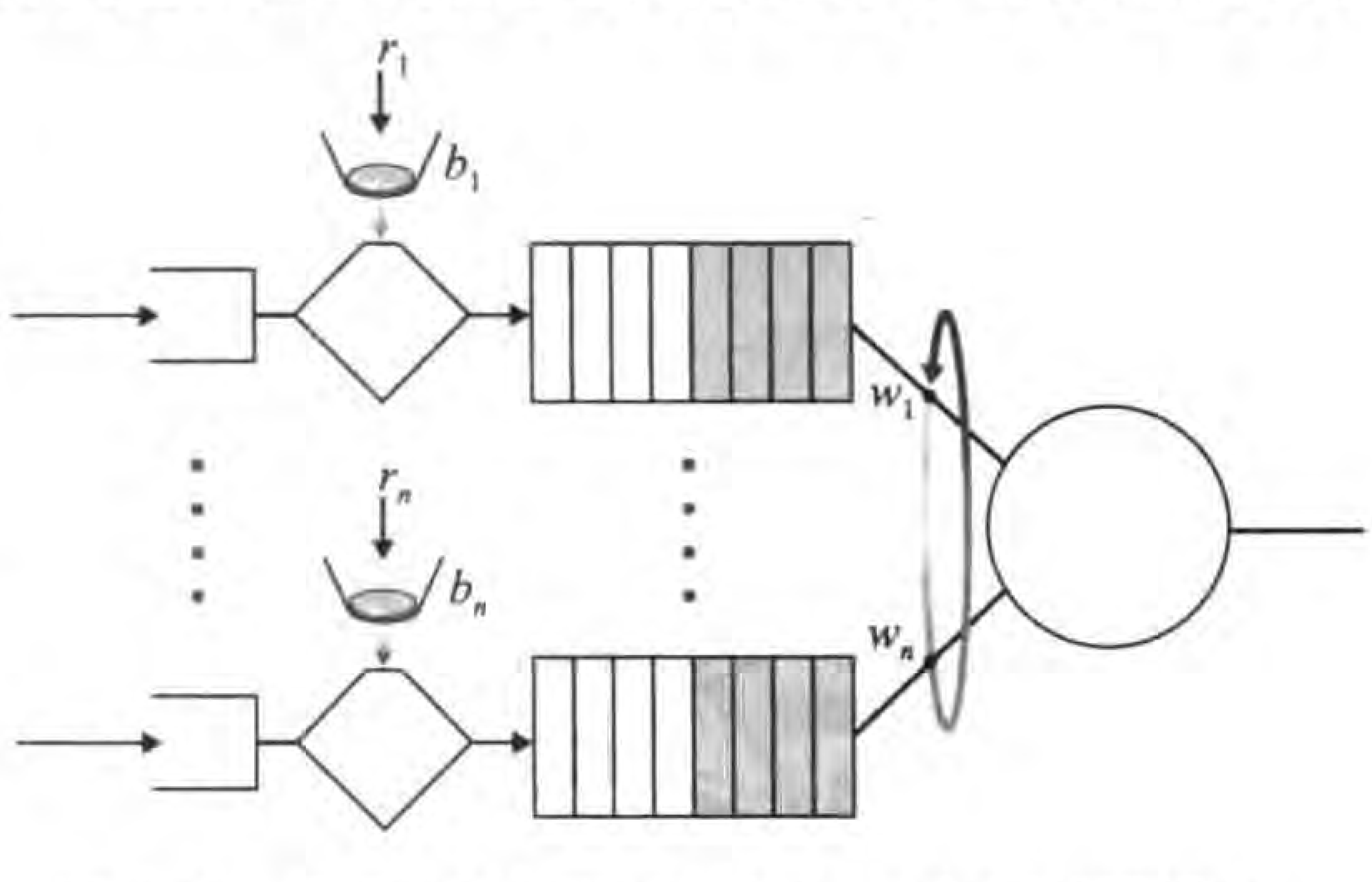


图 9-15 采用 WFQ 调度机制的 n 路复用的漏桶流

我们前面讨论 WFQ 时讲过，每个流 i 保证收到至少等于 $R \cdot w_i/(\sum w_j)$ 的共享链路带宽，其中 R 是以分组/秒为单位的链路传输速率。当以 WFQ 方式等待服务时（也就是通过漏桶传递之后），则分组经受的最大时延是什么？我们来关注流 1。假设流 1 的令牌桶最初是满的。然后 b_1 个分组的突发到达流 1 的漏桶监管器。这些分组去除了漏桶中所有的令牌（没有等待），然后加入了流 1 的 WFQ 等待区域。因为这 b_1 个分组以至少 $R \cdot w_i/(\sum w_j)$ 分组/秒的速度得到服务，直到这些分组的最后一个传输完成，所以将有最大时延 $d_{\max}$，其中

$$d_{\max} = \frac{b_1}{R \cdot \dfrac{w_1}{\sum w_j}}$$

该公式的基本原理是，如果在队列中有 b_1 个分组并且分组以至少每秒 $R \cdot w_i/(\sum w_j)$ 个分组的速率从队列中接受服务（被去除），那么直到最后一个分组的最后一个比特被传送，时间量不会超过 $b_1/(R \cdot w_1/(\sum w_j))$。一道课后习题请你证明，只要 $r_1 < R \cdot w_1/(\sum w_j)$，则 d_{max}确实是流 1 中任何分组在 WFQ 队列中要经受的最大时延。

9.5.3 区分服务

在学习了提供多种类型服务的动机、见解和特定的机制后，我们将注意力集中于以举例方式学习提供多种类型服务的方法，即因特网区分服务（Diffserv）体系结构［RFC 2475；Kilkki 1999］。Diffserv 提供服务区分，也就是在因特网中以一种可扩展性方式用不同的方法处理不同类型流量的能力。可扩展性的需求来源于这样的事实：在因特网的一个主干路由器上同时存在几十万个源到目的地的并行流。我们很快将看到，仅仅通过在网络核心放置简单的功能，而在网络的“边缘”实现更复杂的控制操作，就可以满足该需要。

我们从图 9-16 中显示的简单网络开始讨论。这里我们将描述 Diffserv 的一种可能用法。如 RFC 2475 中所描述，其他的变化是可能的。区分服务体系结构由两个功能元素的集合所组成：

- *边界功能：分组分类和流量调节*。在网络的入边缘（也就是，或者在产生流量的 Diffserv 使能的主机，或者在流量经过的第一个 Diffserv 使能的路由器），到达的分组被标记。更具体地说，在 IPv4 或者 IPv6 分组首部中的区分服务（DS）字段被设置为某个值［RFC 3260］。该 DS 字段的定义意欲取代我们在第 4 章中讨论的 IPv4 服务类型字段和 IPv6 流量类型字段。例如，在图 9-16 中，从 H1 发送到 H3 的分组可能在 R1 被标记，而从 H2 发送到 H4 的分组可能在 R2 被标记。分组得到的标记标识了该分组所属的流量类型。然后不同类型的流量将在核心网络得到不同的服务。

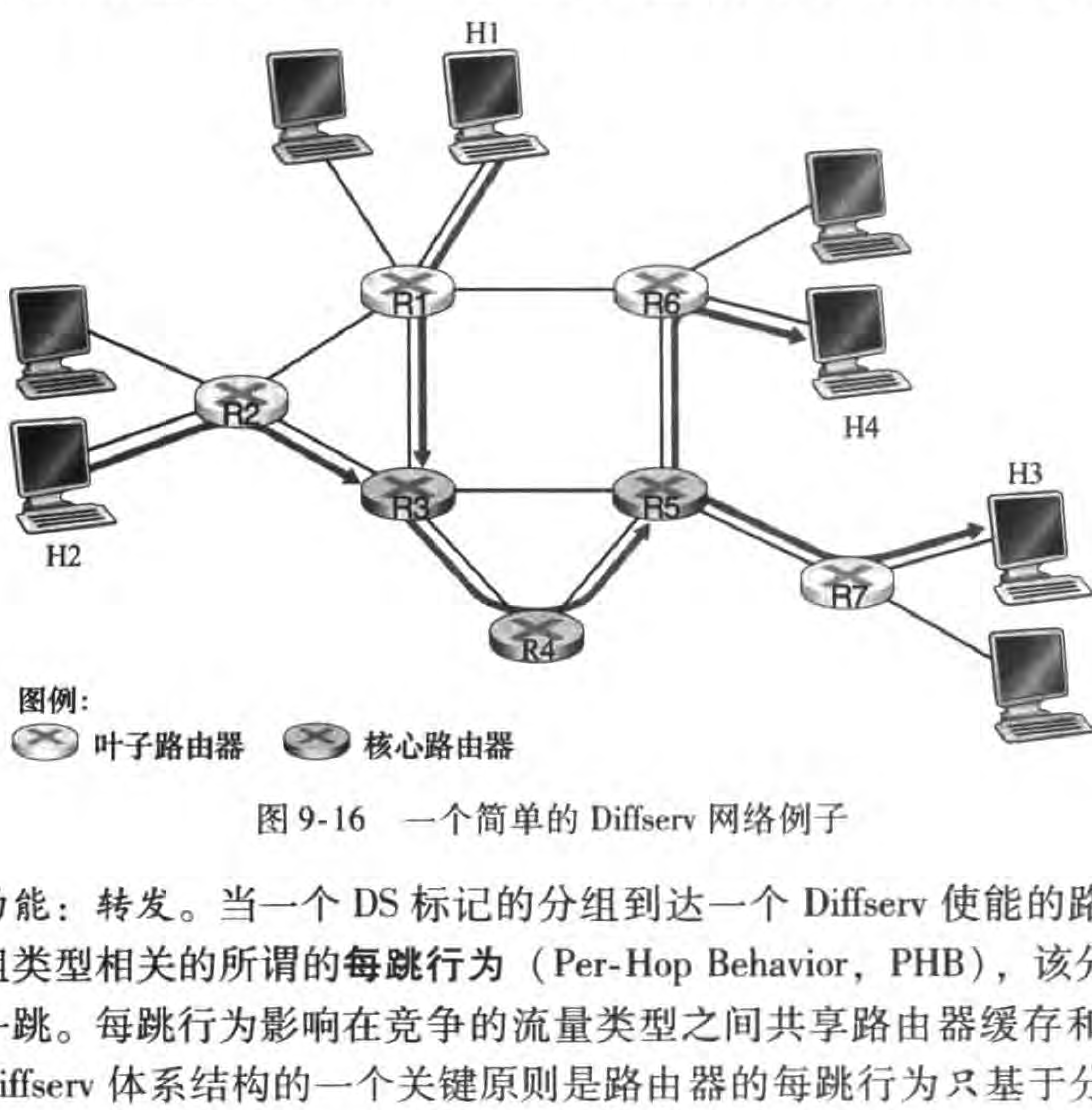


图 9-16 一个简单的 Diffserv 网络例子

- *核心功能：转发*。当一个 DS 标记的分组到达一个 Diffserv 使能的路由器时，根据与分组类型相关的所谓的**每跳行为**（Per-Hop Behavior，PHB），该分组被转发到它的下一跳。每跳行为影响在竞争的流量类型之间共享路由器缓存和链路带宽的方式。Diffserv 体系结构的一个关键原则是路由器的每跳行为*只*基于分组标记，即分

组所属的流量类型。因此，如果图9-16中从H1发送到H3的分组和从H2发送到H4的分组收到同样的标记，网络路由器则将这些分组处理为一个聚合体，而不区别这些分组是源于H1还是源于H2。例如，当这些分组向R4转发时，R3不区分从H1和H2产生的分组。因此，Diffserv体系结构消除了为各个“源到目的地”对而保留路由器状态的要求，这是使Diffserv具有可扩展性的一个重要考虑。

这里有一个类比可能有助于理解相关概念。在很多大规模的社交活动中（例如，一场大型的公众招待会，一场大型的跳舞俱乐部活动或者迪斯科舞会，一场音乐会或者一场足球赛），参加活动的人收到某种类型的入场券。对于非常重要的人物有VIP入场券；对于21岁或者年龄更大的人有21岁以上的入场券（例如是否可以享受酒类饮料）；对于演唱人员有后台入场券；对于记者有新闻入场券；对于普通人有普通的入场券。这些入场券通常在该活动的入口分发，也就是在活动的边缘进行分发。正是在边缘，进行着计算密集型操作，例如交入场费、检查适合的邀请类型以及对照检查邀请与证件。此外，对于允许进入一个活动的特定类型的人数可能有限制。如果有这种限制，人们可能在进入活动之前必须等待。一旦进入了活动，一个人的入场券使他在活动中的很多场所接受有区别的服务，如为VIP提供免费的饮料、较好的桌子、免费食物、单独的房间和殷勤的服务。与之相对照的是，一个普通人禁止进入某些区域，要为饮料支付费用，并且只受到基本服务。在这两种情况下，活动中得到的服务只依赖于入场券的类型。而且在同一类型的所有人得到相同的对待。

图9-17提供了在边缘路由器中分类和标记功能的逻辑视图。到达边缘路由器的分组首先被分类。分类器根据一个或多个分组首部字段的值（例如源地址、目的地址、源端口、目的端口和协议ID）来选择分组，并引导该分组去做合适的标记功能。如前面所指出的，分组的标记携带在分组首部的DS字段中。

在某些情况下，端用户可能已经同意限制其分组发送速率以符合某个申报的**流量配置文件**（traffic profile）。该流量配置文件可能包含对峰值速率和分组流的突发度的限制，如我们前面在漏桶机制中所见。只要用户以符合协商的流量配置文件的方式向网络中发送分组，这些分组就会得到它们的优先级标记，并沿着到目的地的路径转发。另一方面，如果违反了该流量配置文件，那些超出流量配置文件的分组就可能被打上不同的标记，或被整形（例如，为了能够遵守最大速率限制而延时），或可能在网络边缘被丢弃。图9-17中所示的**测定功能**（metering function）的作用是比较进入的分组流和协商的流量配置文件，并确定某分组是否在协商的流量配置文件之内。有关是否立即重新标记、转发、延时或者丢弃一个分组的实际决定是由网络管理员决定的策略问题，而不是由Diffserv体系结构规定的。

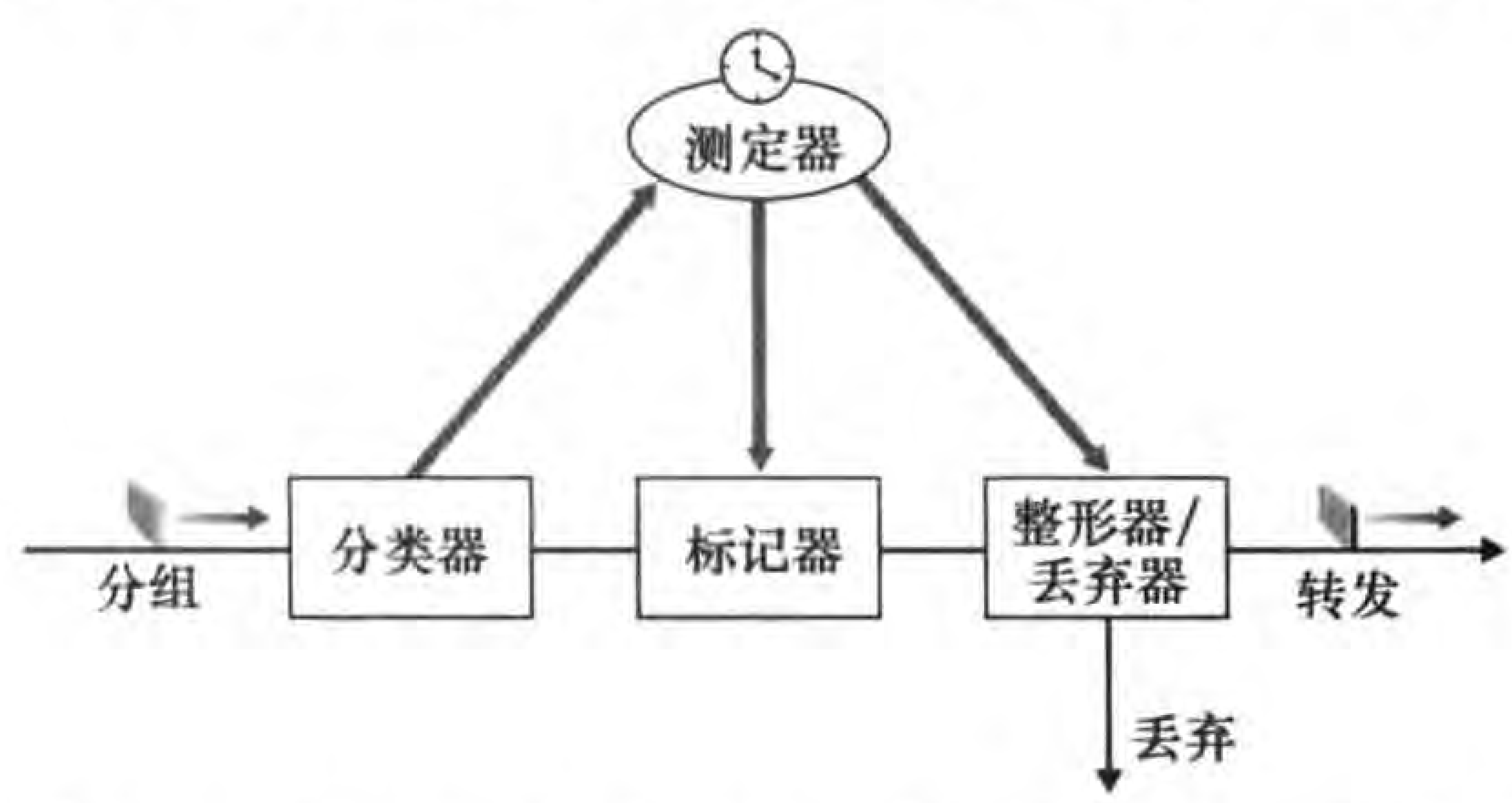


图9-17 在端路由器上的分组分类与流量调节的逻辑视图

到现在为止，我们一直关注着Diffserv体系结构中的标记和监管功能。Diffserv体系结构的第二个关键组件涉及由Diffserv使能路由器所执行的**每跳行为**（PHB）。该每跳行为有点神秘，但被仔细地定义为“Diffserv节点的外部可观察的转发行为的描述，该节点应用了一个特定的Diffserv行为聚合”[RFC 2475]。稍微深入地钻研一下这个定义，我们可以

看到它里面包含的几个重要的考虑：

- PHB 能够导致不同服务类型的流量接收到不同性能（即不同的外部可观察的转发行为）。
- 虽然 PHB 定义了在各种类型之间的性能（行为）差别，但它不强求为了获得这些行为的任何特定机制。只要外部可观察的性能准则得到满足，任何实现机制和任何缓存/带宽分配策略都可以使用。例如，一个 PHB 将不要求使用一个特定的分组排队规则（例如优先级队列、WFQ 队列或 FCFS 队列）以取得特定的行为。PHB 是最后的结果，资源分配和实现机制是达到 PHB 的手段。
- 性能的差别必须是可观察的，因而是可测量的。

已经定义了两种 PHB：一种是加速转发 PHB [RFC 3246]，另一种是确保转发 PHB [RFC 2597]。**加速转发**（Expedited Forwarding，EF）PHB 规定了一类流量离开路由器的速率必须等于或者大于某个已配置的速率。**确保转发**（Assured Forwarding，AF）PHB 将流量分为 4 类，其中每个 AF 类都确保提供某种最小数量的带宽和缓存。

我们通过对 Diffserv 的服务模型进行一些观察来结束对 Diffserv 的讨论。首先，我们隐含地假设 Diffserv 部署在单个管理域中，但典型情况是端到端的服务必须在跨越通信端系统之间的多个 ISP 来形成。为了提供端到端的 Diffserv 服务，端系统之间的所有 ISP 不仅必须提供这种服务，而且为了向端用户提供真正的端到端服务，多数 ISP 之间也要协作并做出安排。如果没有这种协作，直接向客户出售 Diffserv 的 ISP 会发现他们将重复地说："是的，我们知道你支付了额外费用，但是我们与丢弃和延迟你流量的 ISP 之间没有服务约定。我很抱歉在你的 VoIP 电话中有很多间隙！"第二，如果 Diffserv 实际存在并且该网络运行的负载不大，大多数时间尽力而为服务和 Diffserv 服务之间将没有可觉察的差异。的确，端到端时延通常是由接入速率和路由器跳数造成的，而不是由路由器中的排队时延造成的。想象一个支付了增值服务费用的不幸的 Diffserv 客户，发现为其他人提供的尽力而为服务几乎总是具有与增值服务相同的性能！

9.5.4 每连接服务质量保证：资源预约和呼叫准入

在前一节中，我们已经看到了分组标记和监管、流量隔离以及链路级调度能够为一类服务提供比另一类更好的性能。在某些调度规则下，如优先权调度，较低类型的流量基本对最高优先权类型的流量是"不可见"的。借助于适当的网络定制，最高类型的服务的确能够取得极低的丢包和时延，即基本上是类似于电路的性能。但是，这种网络能够确保一个在高优先权流量类型中的进行流，仅使用我们已经描述的这些机制就能在整个流期间持续得到这样的服务吗？答案是否定的。在本节中，我们将看到当对各个连接提供硬服务保证时，为什么还需要另外的网络机制和协议。

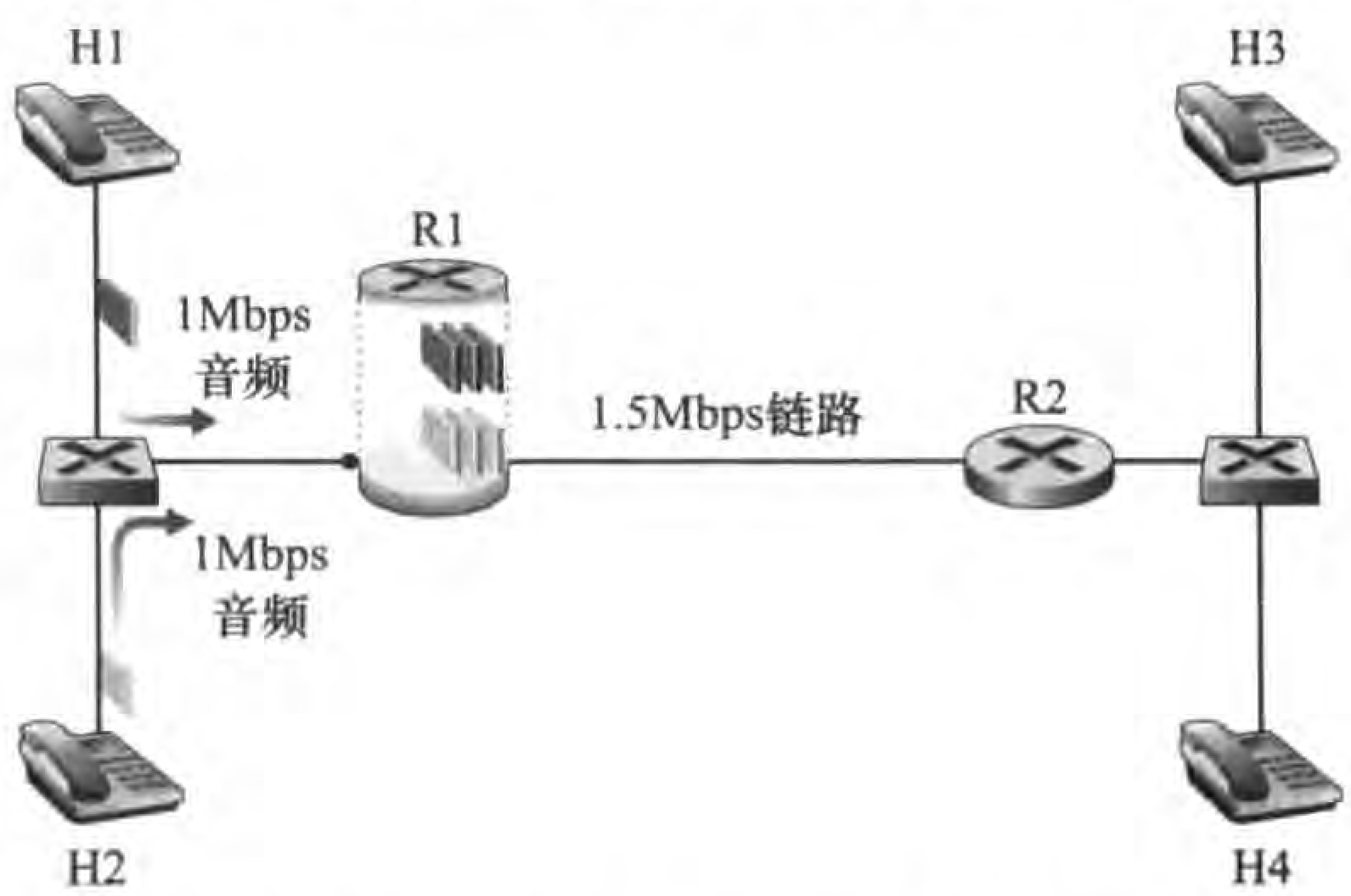


图 9-18　两个竞争的音频应用过载 R1 到 R2 的链路

我们返回 9.5.2 节的场景中，考虑两个 1Mbps 的音频应用经 1.5Mbps 的链路传输它们的分组，如图 9-18 所示。这两

条流的结合数据率（2Mbps）超过了这条链路的容量。即使使用分类和标记、流量隔离以及共享未使用的带宽（这里并没有未使用的带宽），这很明显还是一个与丢包有关的命题。只是因为没有足够的带宽同时满足两个应用的需求。如果这两个应用平等地共享该带宽，每个应用将会丢失其25%的传输分组。这是一个无法接受的低QoS，以至于两个音频应用完全不可用；甚至从一开始就没有必要传输任何音频分组。

在图9-18中两个应用不能同时满足的情况下，这个网络应当做些什么呢？如果允许两者以一种不可用的QoS继续在应用流上浪费网络资源，最终将无法向端用户提供任何效用。希望该答案是清晰的，即应用流中的一个应当被阻塞（即拒绝接入网络），而另一个应当允许继续进行，使该应用使用所需的全部1Mbps。电话网是一个执行这种呼叫阻塞的网络例子，即如果要求的资源（在电话网的情况下是一个端到端的电路）不能分配给该呼叫，该呼叫就被阻塞了（阻止进入网络），并且返回给用户一个忙信号。在我们的例子中，如果一个流没有分配到足够的QoS来使自己可用，允许它进入网络没有任何好处。的确，接纳一个不能得到它需要的QoS的流是要付出代价的，因为网络资源被用于支持一个对端用户没有用的流。

通过基于流的资源要求以及已经准入流的资源要求明确地准入或阻塞流，网络能够保证准入流可得到它们所请求的QoS。这隐含的意思是：需要为流提供有保证的QoS就需要流来申告它的QoS需求。让流申告它的QoS需求，然后让网络接受该流（以所要求的QoS）或者阻塞该流的过程称为**呼叫准入**（call admission）过程。这则是我们的第4个见解（除了前面9.5.2节中的3个见解外），该见解增加了提供QoS所需的机制。

见解4：如果不总是能够提供充分的资源，并且要确保QoS，则需要一个呼叫准入过程，在这个过程中流申告其QoS需求，然后要么被网络准入（以所要求的QoS），要么被网络阻塞（如果网络不能提供所要求的QoS）。

如果一个呼叫（一个端到端流）一旦开始就要被确保给定的服务质量，则我们在图9-18中的促进思考的例子强调了对几种新的网络机制和协议的需求：

- 资源预留。为了满足一个呼叫所希望的QoS，确保它具有所需的资源（链路带宽、缓存）的唯一方法是显式地为该呼叫分配这些资源，用网络的行话来讲，该过程称为**资源预留**（resource reservation）。一旦呼叫预约了资源，它在其整个过程中按需访问这些资源，而不管所有其他呼叫的需求。如果一个呼叫预约并得到链路带宽的xMbps的保证，而且传输速率决不会大于x Mbps，则该呼叫将具有无丢包和无时延的性能。
- 呼叫准入。如果预留了资源，则该网络必须具有一种用于呼叫请求和预留资源的机制。由于资源不是无限的，如果请求的资源不可用，则进行呼叫准入请求的呼叫将被拒绝准入，即被阻塞。电话网执行的是这种呼叫准入机制，即当我们拨一个号码时，就请求了资源。如果完成该呼叫所需的电路（TDMA时隙）是可用的，则分配电路并且完成了呼叫。如果电路不可用，则该呼叫被阻塞，我们得到了忙信号。为了得到网络的准入，被阻塞的呼叫能够一再尝试，但是直到它成功地完成呼叫准入过程，才被允许向网络发送流量。当然，分配链路带宽的路由器，其分配的带宽不应当超过链路可用的带宽。通常，一个呼叫仅能预约一条链路的带宽的一部分，因此一台路由器可以为多于一条呼叫分配链路带宽。然而，如果要提供硬服务质量保证的话，为所有呼叫分配的带宽总和应当小于该链路的容量。
- 呼叫建立信令。以上描述的呼叫准入过程要求一条呼叫沿着它的源到目的地路径

在每台网络路由器中能够预约充足的资源，以确保满足它的端到端 QoS 请求。每台路由器必须决定会话所请求的本地资源，考虑已经承诺给其他进行中的会话的资源量，并决定它是否有足够的资源来满足在这台路由器上的该会话的每跳 QoS 要求，而不违反对所有已经准入的会话做出的本地 QoS 保证。需要一个信令协议来协调这些各种各样的活动，即本地资源的每跳分配，以及该呼叫是否能够在沿着端到端路径上的每台路由器中预约充分的资源。这是如图 9-19 所示的**呼叫建立协议**（call setup protocol）的任务。**资源预留协议**（resource ReSerVation Protocol，RSVP）[Zhang 1993，RFC 2210] 就是在因特网体系结构中为此目的所提出的，用于提供服务质量保证。在 ATM 网络中，Q2931b 协议 [Black 1995] 在 ATM 网络的交换机和端点之间承载了这种信息。

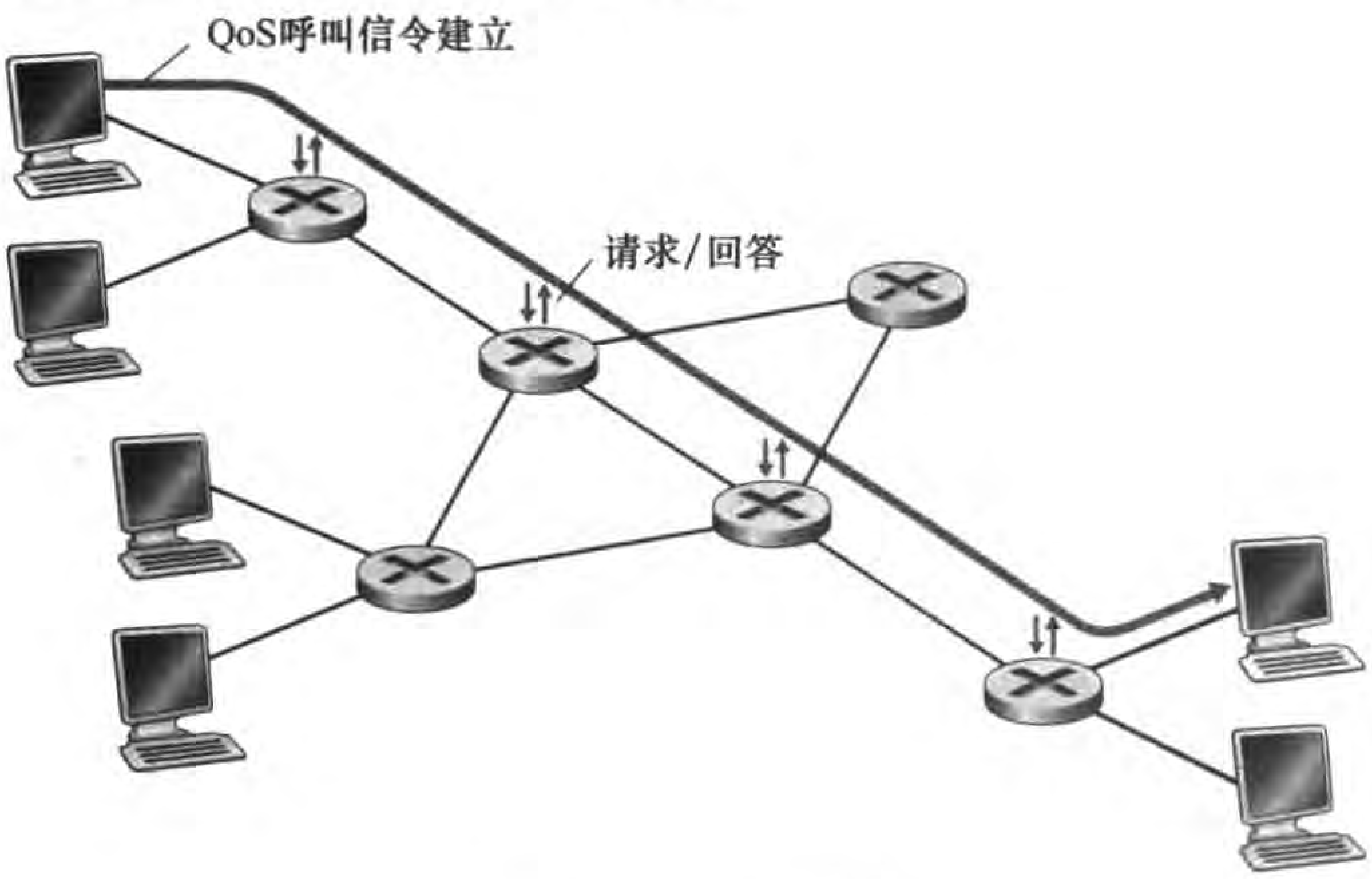


图 9-19 呼叫建立过程

尽管投入了巨大的研究和开发力量，并且甚至开发出了提供每连接服务质量保证的产品，但是几乎没有这种服务的扩展部署。有许多可能的原因。首当其冲的是，情况很可能是我们在 9.2 ~ 9.4 节学习的简单的应用级机制与适当的网络定制相结合（9.5.1 节），为多媒体应用提供了“足够好的”尽力而为的网络服务。此外，ISP 判断：提供每连接服务质量保证所需的附加复杂性以及部署和管理网络的成本，相对于该服务可预测的用户收入，前者可能过高。

9.6 小结

多媒体网络是当今因特网中最为激动人心的发展之一。全世界的人们在收音机和电视机面前花费的时间越来越少，而转向使用他们的智能手机和设备接收音频和视频传输，它们包含了实况转播和预先录制的节目。此外，借助于如 YouTube 这样的网站，用户除了是多媒体因特网内容的消费者，也成为内容的生产者。除了视频分发，因特网也正在用于传输电话。事实上，经过下一个 10 年，因特网连同无线因特网接入，可能致使传统的电路交换电话系统成为明日黄花。VoIP 不仅提供了廉价的电话服务，而且将提供大量的增值服务，例如视频会议、在线目录服务、语音消息及融入如脸书和微信等社交网络。

在 9.1 节中，我们描述了视频和语音的内在特点，将多媒体应用分为 3 种类型：(i) 流式存储音频/视频；(ii) 会话式 IP 语音/视频；(iii) 流式实况音频/视频。

在9.2节中，我们更为深入地学习了流式存储视频。对于流式视频应用，预先录制的视频放置在服务器上，用户向这些服务器发送请求以按需观看这些视频。我们看到流式视频系统能够被分为两种类型：UDP流和HTTP流。我们观察到对于流式视频的最为重要的性能测量是平均吞吐量。

在9.3节中，我们研究了诸如VoIP等会话式多媒体应用能够设计运行在尽力而为网络上的方法。对于会话式多媒体，定时考虑是重要的，因为会话式应用是高度时延敏感的。在另一方面，会话式多媒体应用是容忍丢包的，即偶尔的丢包仅会在音频/视频播放中引起偶尔的干扰信号，并且这些丢包常常能被部分地或者全部地掩盖。我们看到将客户缓存、分组序号和时间戳结合起来的方法，极大地缓解了网络引入的时延抖动的影响。我们也展望了Skype所依赖的技术，Skype是IP语音和IP视频界的领头公司之一。在9.4节中，我们学习了两种最为重要的用于VoIP的标准化协议，即RTP和SIP。

在9.5节中，我们介绍了几种网络机制（链路级调度规则和流量监管）是如何用于在几类流量之间提供区分服务的。

课后习题和问题

复习题

9.1节

R1. 针对Victor Video正在观看4Mbps视频，Facebook Frank每20秒观看一幅新的100KB图片，以及Martha Music正在听200kbps音频流的情况，重新构造9-1表。

R2. 在视频中有两种类型的冗余。描述它们，并讨论如何能够利用它们进行有效压缩。

R3. 假定一个模拟音频信号每秒抽样16 000次，并且每个样本量化为1024级之一，该PCM数字音频信号的比特率将是多少？

R4. 多媒体应用能够分为三种类型。阐述它们的名称并对每种类型进行描述。

9.2节

R5. 流式多媒体系统能够分为三种类型。阐述它们的名称并对每种类型进行描述。

R6. 列举UDP流的三种缺点。

R7. 对于HTTP流，TCP接收缓存和客户应用缓存是相同的东西吗？如果不是，它们是怎样交互的呢？

R8. 考虑对于HTTP流的简单模型。假设服务器以2Mbps的恒定速率发送比特，并且当已经接收到800万比特时开始播放。初始缓存时延 t_p 是多少？

9.3节

R9. 端到端时延和分组时延抖动的区别是什么？分组时延抖动的原因是什么？

R10. 为什么在预定的播放时间之后收到的分组被认为是丢失了？

R11. 9.3节描述了两种FEC方案，简要地总结它们。这两种方案通过增加开销而增加了流的传输速率。交织技术也会增加传输速率吗？

9.4节

R12. 接收方如何识别在不同会话中的不同RTP流？同一个会话中的不同流是怎样识别的？

R13. SIP注册器的作用是什么？一个SIP注册器的作用与移动IP中的归属代理的作用有怎样的差异？

习题

P1. 考虑下图。类似于我们对图9-1的讨论，假设视频以固定比特率进行编码，因此每个视频块包含了在相同的固定时间量Δ播放的视频帧。服务器在 t_0 时刻传输第一个视频块，在 $t_0+\Delta$ 时刻传输第二

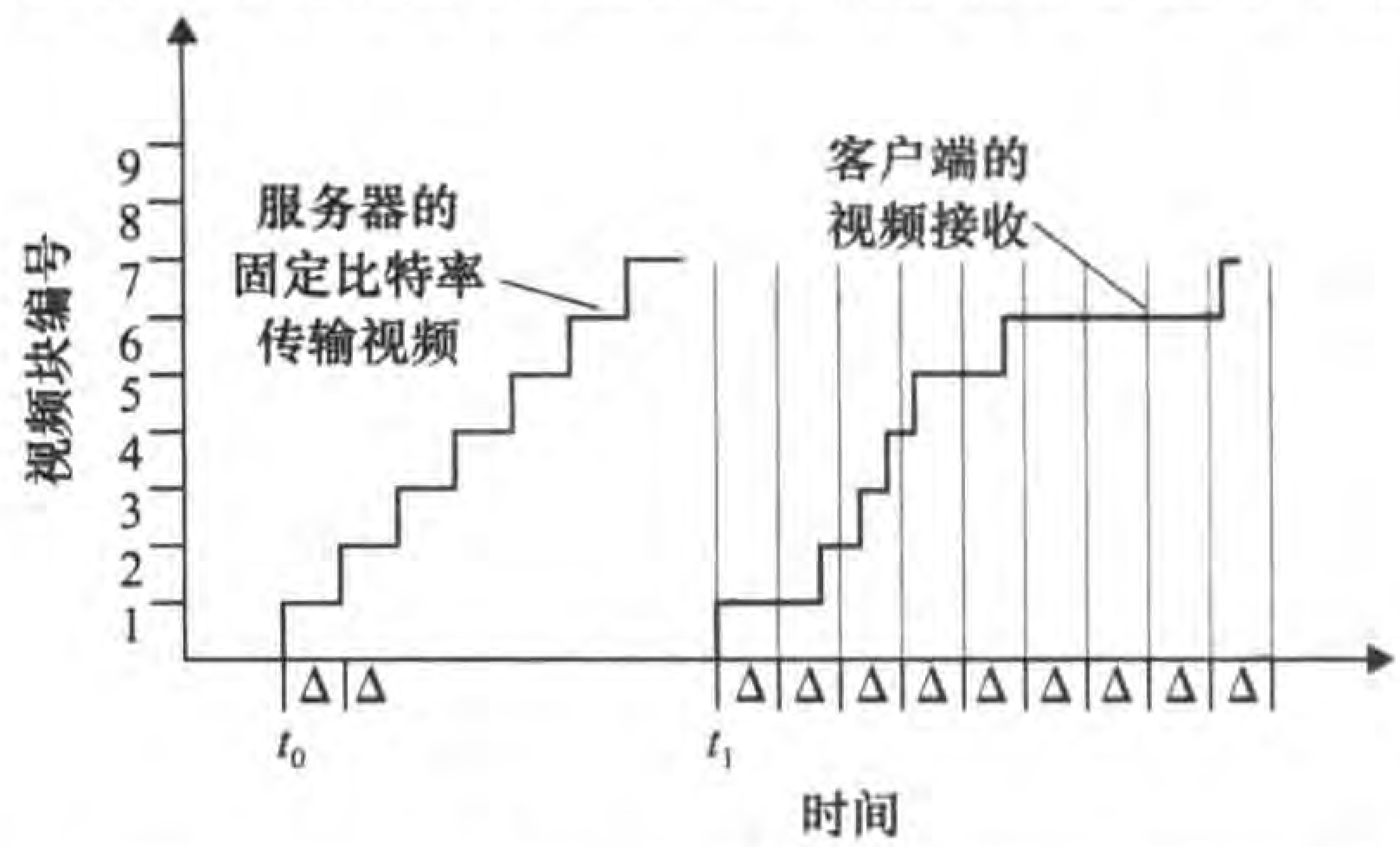
块，在 $t_0+2\Delta$ 时刻传输第三块，等等。一旦客户开始播放，每个块应当在前一块后播放 Δ 时间单元。

a. 假设第一块在 t_1 时刻一到达，客户就开始播放。在下图中，多少视频块（包括第一块）将及时到达客户供其播放？解释你怎样得到该答案的。

b. 假设该客户现在在 $t_1+\Delta$ 时刻开始播放。多少视频块（包括第一块）将及时到达客户供其播放？解释你怎样得到该答案的。

c. 在如上面（b）的相同场景中，存储在客户缓存中等待播放的块的最大数量有多少？解释你怎样得到该答案的。

d. 在客户中为使每个视频块及时到达供其播放，最小的播放时延是多少？解释你怎样得到该答案的。



P2. 回想在图9-3中显示的HTTP流的简单模型。前面讲过 B 表示客户应用缓存的长度，Q 表示在客户应用缓存开始播放之前必须被缓存的比特数量。还有 r 表示视频消耗速率。假设无论何时客户缓存不满，服务器都以恒定速率 x 发送比特。

a. 假设 $x<r$。如在本书所讨论，在此场合播放将在连续播放期和停滞期之间变动。作为 Q、r 和 x 的函数，确定每个连续播放和停滞期的长度。

b. 现在假设 $x>r$。求客户应用缓存变满的时刻 $t=t_f$？

P3. 回想在图9-3中显示的HTTP流的简单模型。假设缓存长度是有限的，并服务器以可变的速率 $x(t)$ 发送比特。具体而言，假设 $x(t)$ 具有下列锯齿形状。其速率在时刻 $t=0$ 初始为零，在时刻 $t=T$ 线性爬升到 H。然后反复重复这种模式，如下图中显示的那样。

a. 服务器的平均发送速率是什么？

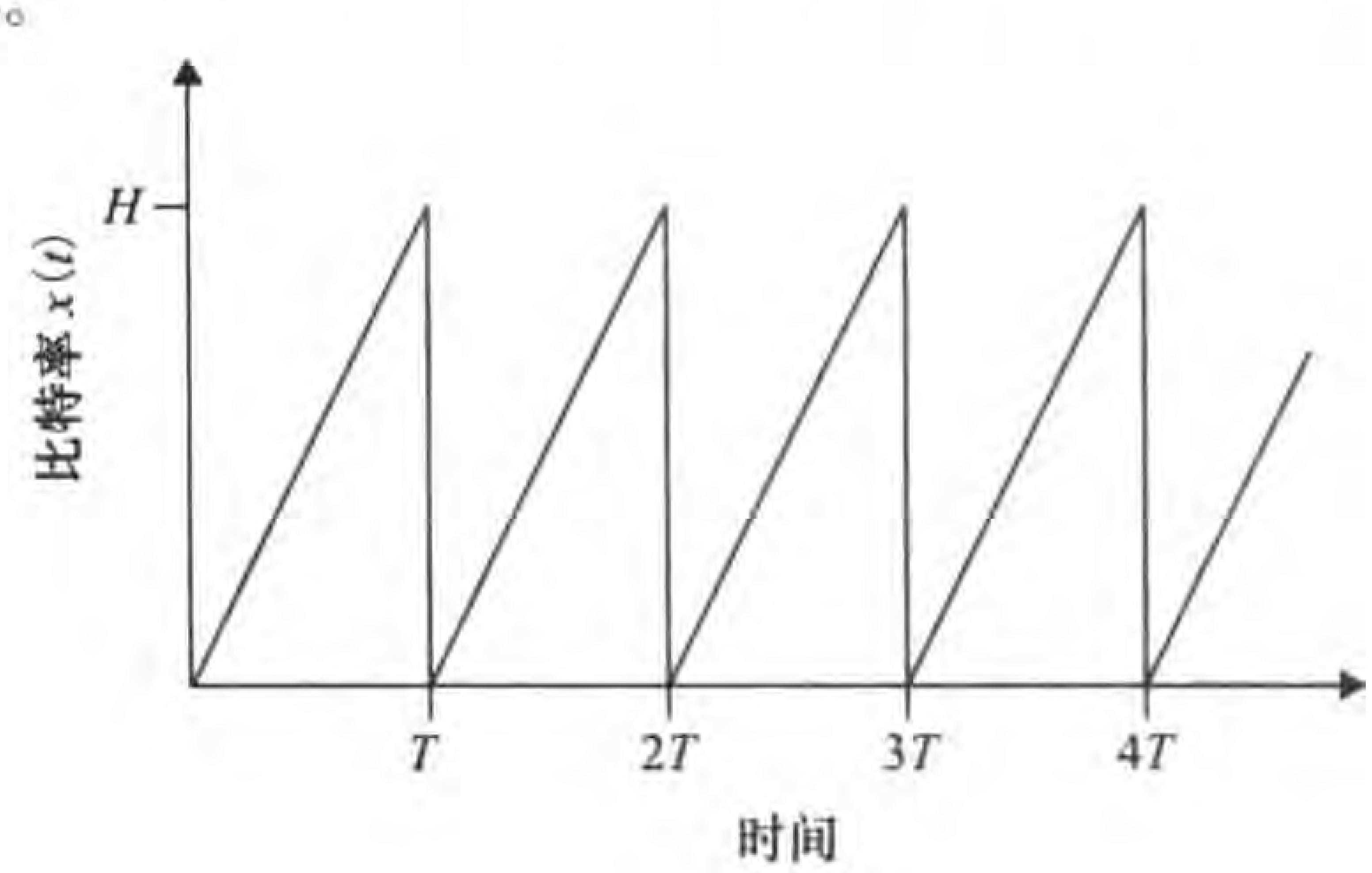
b. 假设 $Q=0$，客户一接收到一个视频块就开始播放。将发生什么现象？

c. 现在假设 $Q>0$。作为 Q、H 和 T 的函数，确定首先开始播放的时间。

d. 假设 $H>2r$ 和 $Q=HT/2$。证明在初始播放时延之后将不会有停滞。

e. 假设 $H>2r$。求出最小的 Q 值，使在初始播放时延之后将不会有停滞。

f. 现在假定缓存长度 B 是有限的。假定 $H>2r$。作为 Q、B、T 和 H 的函数，确定当客户应用缓存首次变满的时刻 $t=t_f$。



P4. 回想在图 9-3 中显示的 HTTP 流的简单模型。假设客户应用缓存长度是有限的，服务器以恒定的速率 x 发送，并且视频的压缩率是 r，其中 $r < x$。还假设播放立即开始。假设用户早在时刻 $t = E$ 中止视频。在中止的时刻，服务器停止发送比特（如果它还没有发送视频中所有比特的话）。

a. 假设视频为有限长。问浪费了多少比特（即发送但未观看）？

b. 假设视频为 T 秒长，其中 $T > E$。问浪费了多少比特（即发送但未观看）？

P5. 考虑一个 DASH 系统，其中有 N 个视频版本（有 N 种不同的速率和质量）和 N 种音频版本（有 N 种不同的速率和质量）。假设我们要允许播放者在任何时间对任何 N 种视频版本和任何 N 种音频版本进行选择。

a. 如果我们生成文件，使音频与视频混合起来，这样服务器在给定的时间仅发送一种媒体流，该服务器将需要存储多少文件（每个文件有一种不同的 URL）？

b. 如果服务器单独发送音频流和视频流并且让客户同步这些流的话，该服务器将需要存储多少文件？

P6. 在 9.3 节的 VoIP 例子中，令 h 为加到每个块中的首部字节总数，包括 UDP 和 IP 的首部。

a. 假设每 20 秒发射一个 IP 数据报，求出由该应用一侧产生的数据报的每秒传输速率（以比特计）。

b. 当使用 RTP 时，h 的典型值是什么？

P7. 考虑在 9.3 节中描述估计平均时延 d_i 的过程。假设 $u = 0.1$。令 $r_1 - t_1$ 是最近的采样时延，令 $r_2 - t_2$ 是下一个最近的采样时延，等等。

a. 对于一个特定的音频应用，假设四个分组到达接收方的采样时延为 $r_4 - t_4$、$r_3 - t_3$、$r_2 - t_2$ 和 $r_1 - t_1$。根据这 4 个采样来表示时延 d 的估计值。

b. 对于 n 个采样时延归纳出公式。

c. 对于 b 小题的公式，令 n 趋于无穷，给出最后的公式。评论一下为什么这个平均过程被称为一个指数移动平均数。

P8. 重复习题 P7 中的（a）和（b），求出平均时延偏差的估计值。

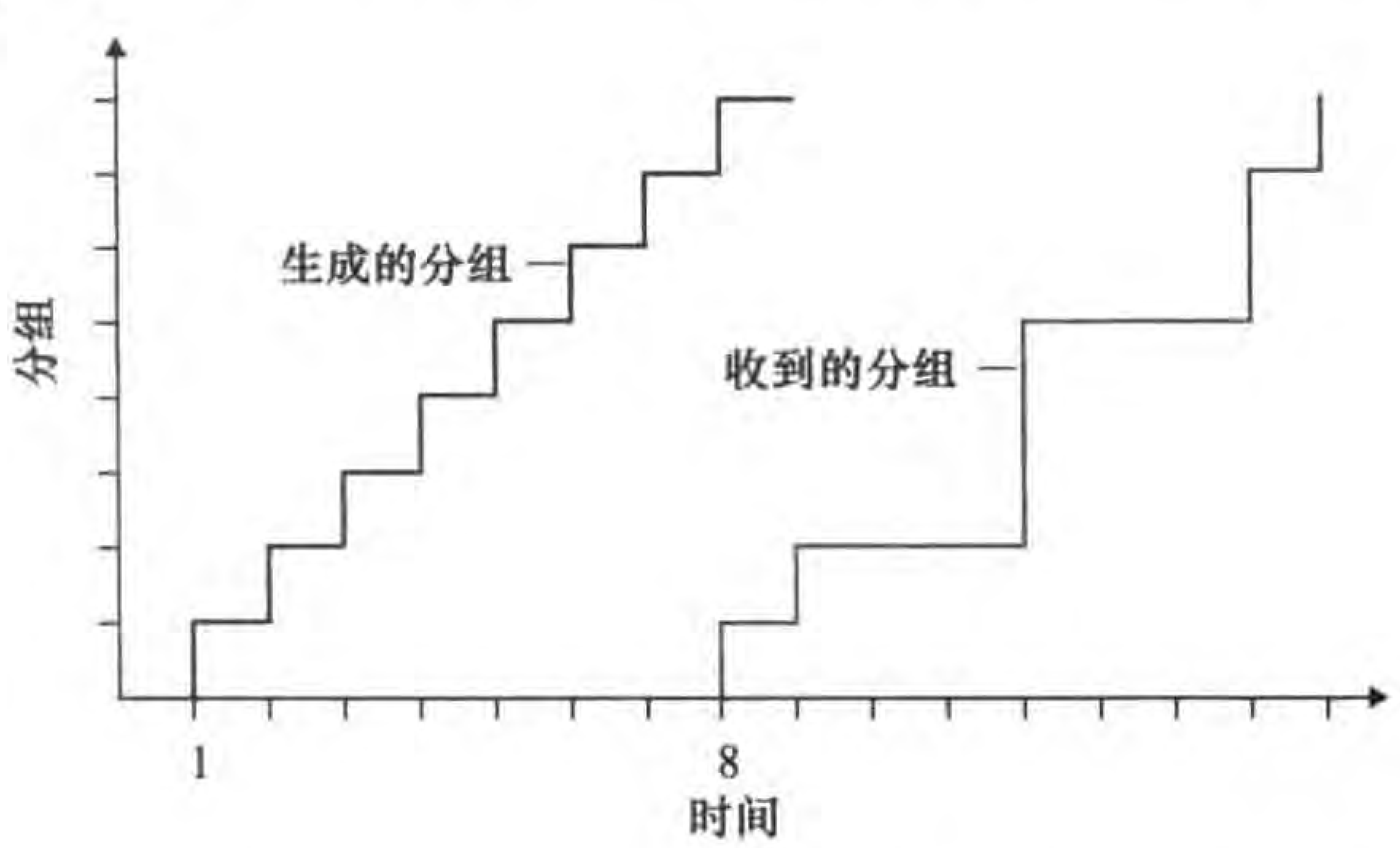
P9. 对于 9.3 节中的 VoIP 例子，我们引入了一个估计时延的在线过程（指数移动平均数）。在本习题中我们将研究另一种过程。设 t_i 是接收到的第 i 个分组的时间戳；设 r_i 是收到第 i 个分组的时间。令 d_n 表示在收到第 n 个分组后我们对平均时延的估计。在收到第一个分组后，我们设置时延估计值等于 $d_1 = r_1 - t_1$。

a. 假设我们希望对于所有的 n，有 $d_n = (r_1 - t_1 + r_2 - t_2 + \cdots + r_n - t_n)/n$。根据 d_{n-1}、r_n 和 t_n，给出 d_n 的递归公式。

b. 描述一下对于 VoIP，为什么在 9.3 节中描述的时延估计比（a）部分叙述的时延估计更合适。

P10. 比较 9.3 节描述的估计平均时延的过程和 3.5 节估计往返时间的过程。这些过程有什么相同？有什么不同？

P11. 考虑下图（它类似于图 9-3）。某发送方在 $t = 1$ 时开始周期性地发送分组化的音频。在 $t = 8$，第一个分组到达接收方。



a. 分组 2 到分组 8 的时延（从发送方到接收方，忽略任何播放时延）是什么？注意到在该图上每个

垂直和水平线段具有1、2或3个时间单元的长度。

b. 如果第一个分组在 $t=8$ 时刻一到达接收方，音频就开始播放，发送的前8个分组中的哪些将不能按时到达进行播放？

c. 如果在 $t=9$ 时刻音频开始播放，发送的前8个分组中的哪些将不能按时到达进行播放？

d. 在接收方，导致所有前8个分组按时到达进行播放的最小播放时延是什么？

P12. 再次考虑习题P11中的图，显示分组音频传输和接收时间。

a. 计算分组2到分组8的估计时延，使用9.3.2节中的对于 d_i 的公式。使用 $u=0.1$ 的值。

b. 使用9.3.2节中对于 v_i 的公式，从对分组2到分组8的估计平均时延计算其估计的偏差。

P13. 在9.3节中讲过VoIP的两种FEC方案。假设第一种方案为每4个初始块产生一个冗余块。假设第二种方案使用传输速率为标称流传输速率的25%的低比特率编码。

a. 每种方案需要多少额外带宽？每种方案增加多少播放时延？

b. 如果在每组的5个分组中第一个分组丢失了，这两种方案如何执行？哪一种方案有更好的音频质量？

c. 如果在每组的2个分组中第一个分组丢失了，这两种方案如何执行？哪一种方案有更好的音频质量？

P14. a. 考虑Skype中的一个音频会议具有 $N>2$ 个与会者。假定每个与会者生成一个速率 $r$bps 的恒定流。呼叫的发起方将需要每秒发送多少比特？其他 $N-1$ 个与会者每个都将需要每秒发送多少比特？聚合所有与会者，总的发送速率是多少？

b. 对于使用一台中心服务器的一个Skype视频会议，重复（a）。

c. 重复（b），只是现在情况为每个对等方发送其视频流副本到其他 $N-1$ 个对等方。

P15. a. 假设我们向因特网发送两个IP数据报，每个数据报携带不同的UDP段。第一个数据报的源IP地址为A1，目的IP地址为B，源端口为P1，目的端口为T。第二个数据报的源IP地址为A2，目的IP地址为B，源端口为P2，目的端口为T。假设A1和A2不同，P1和P2不同。假设这两个数据报都到达它们的目的地址，这两个UDP数据报会被同一个套接字接收吗？为什么？

b. 假设Alice、Bob和Claire要使用SIP和RTP来进行音频会议呼叫。Alice与Bob和Claire之间发送和接收RTP分组，只有一个UDP套接字足够吗（SIP报文所需的套接字除外）？如果够，那么Alice的SIP客户如何区分RTP分组是来自Bob还是来自Claire？

P16. 是非判断题：

a. 如果存储视频直接从Web服务器流式传输到媒体播放器，这个应用则正在使用TCP作为底层的传输协议。

b. 当使用RTP时，发送方有可能在会话的中间改变编码。

c. 所有使用RTP的应用必须使用端口87。

d. 假设一个RTP会话对每个发送方有独立的音频和视频流，则这些音频和视频流使用同样的SSRC。

e. 在区分服务中，尽管每跳行为定义了各类型之间的性能差别，但它没有强制要求为了获得这些性能而使用任何特定机制。

f. 假设Alice要和Bob建立一个SIP会话。在她的INVITE报文中包括了这样的行：m = audio 48753 RTP/AVP 3（AVP 3指示GSM音频）。因此Alice在该报文中指示她要发送GSM音频。

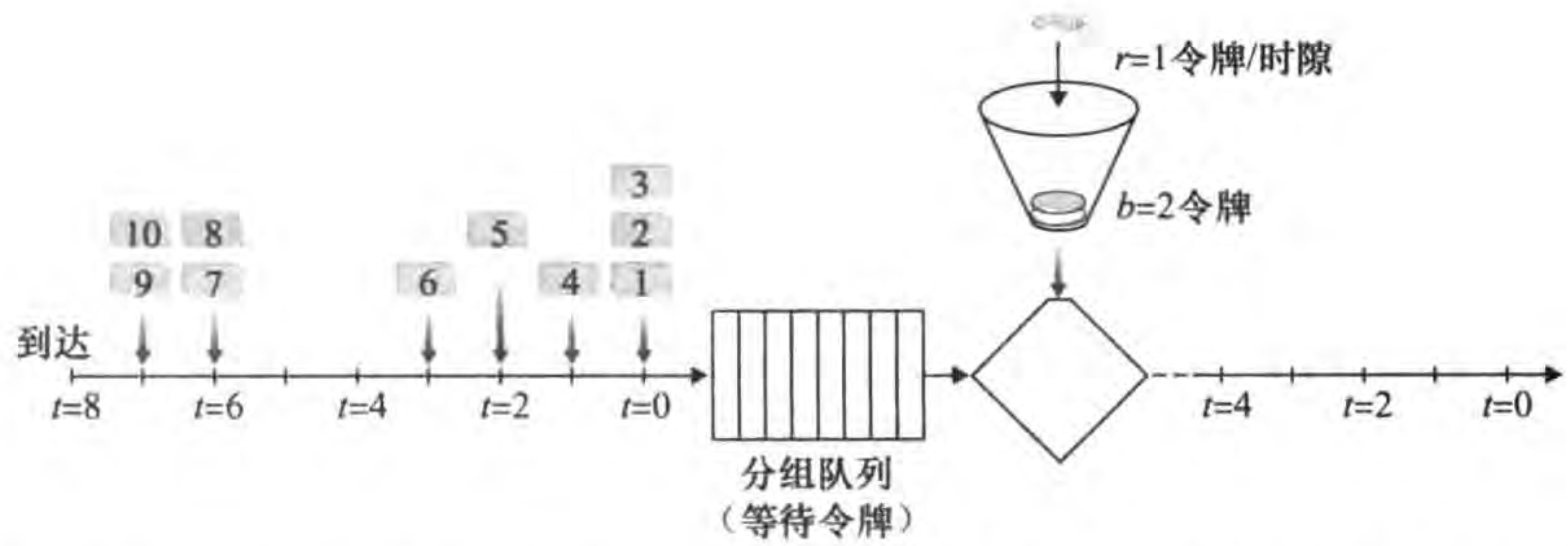
g. 参考前一句说法，Alice在她的INVITE报文中指示了她将把音频发送到端口48753。

h. SIP报文在SIP实体之间通常使用一个默认的SIP端口号发送。

i. 为了维护注册，SIP客户必须周期地发送REGISTER报文。

j. SIP强制所有的SIP客户支持G.711音频编码。

P17. 考虑下图，图中显示了一个由分组流提供输入的漏桶监管器。令牌桶能够保持最多两个令牌，并最初在 $t=0$ 时刻是满的。新的令牌以每时隙1个令牌的速率到达。输出链路速率采用下列规则：如果两个分组在一个时隙的开始获得令牌，它们能够在相同的时隙中进入输出链路。该系统的定时细节如下：



1. 分组在时隙的开始到达（如果有的话）。因此在该图中，分组 1、2 和 3 在时隙 0 到达。如果在队列中已经有分组，则到达的分组加入该队列的尾部。分组以 FIFO 方式朝着该队列的前方行进。
2. 当到达分组增加进队列之后，如果有任何排队分组，那些分组中的一个或两个（取决于可用令牌的数量）将每个从令牌桶中去除一个令牌，并在那个时隙去往输出链路。因此，分组 1 和 2 从桶中去除一个令牌（因为最初有两个令牌）并在时隙 0 期间去往输出链路。
3. 如果令牌桶未满则新令牌加入，因为令牌产生速率是 $r=1$ 令牌/时隙。
4. 然后时间前进到下一个时隙，并且重复这些步骤。

回答下列问题：

a. 对每个时隙，在到达分组处理后（上述步骤 1）但在任何分组通过队列传输并去除一个令牌之前的那个时刻，指出位于队列中的分组和位于桶中的令牌数量。因此，对于在上述例子中的 $t=0$ 时刻的时隙，分组 1、2 和 3 位于队列中，桶中有两个令牌。

b. 对每个时隙，指出在令牌从队列中去除后哪个分组出现在输出链路。这样，对于上述例子中的 $t=0$ 时隙，在时隙 0 期间，分组 1 和 2 从漏桶出现在输出链路上。

P18. 重复习题 P17，但假设 $r=2$。再次假设初始时桶是满的。

P19. 考虑习题 P18 并假设现在 $r=3$ 且 $b=2$ 与以前一样。你对上述问题的回答有变化吗？

P20. 考虑一下监管分组流的平均速率和突发长度的漏桶监管器。我们现在也要监管峰值速率 p。说明这个漏桶监管器的输出如何能够提供给第二个漏桶监管器，使得这两个串行的漏桶能够监管平均速率、峰值速率和突发长度。要给出第二个监管器的桶长度和令牌产生速率。

P21. 如果对于任何 t 来说，在每个时间间隔长度 t 内到达漏桶的分组数小于 $rt+b$ 个，那么就说分组流符合一个突发长度 b 和平均速率 r 的漏桶规范 (r, b)。一个符合漏桶规范 (r, b) 的分组流必须在参数 r 和 b 的漏桶监管器那里等待吗？评估你的答案。

P22. 说明只要 $r_1 < Rw_1/(\sum w_j)$，那么 d_{max} 实际上是流 1 中任何分组在 WFQ 队列中要经受的最大时延。

编程作业

在这个实验中，将实现流式视频服务器和客户。该客户使用实时流协议（RTSP）来控制服务器的动作。服务器使用实时协议（RTP）来分组化视频以便在 UDP 上传输。将为你提供在客户和服务器中部分实现了 RTSP 和 RTP 的 Python 代码，你的工作是完成客户和服务器代码。当完成时，你将已经创建了一个进行下面工作的客户 - 服务器应用：

- 客户发送 SETUP、PLAY、PAUSE 和 TEARDOWN 等 RTSP 命令，并且服务器应答这些命令。
- 当服务器处于播放状态，它周期地抓取存储的 JPEG 帧，用 RTP 对该帧分组化，并将该 RTP 分组发送到一个 UDP 套接字中。
- 客户接收该 RTP 分组，去除 JPEG 帧，解压缩该帧，并在客户的监管器上再现该帧。

为你提供的代码在服务器实现 RTSP 协议，在客户实现对 RTP 解分组化。这个代码也考虑传输视频的显示。你需要在客户实现 RTSP 和 RTP 服务器。该编程作业将极大地增强学生对 RTP、RTSP 和流式视频的理解。我们极力推荐它。该作业也建议进行大量可选的练习，包括在客户和服务器实现 RTSP 的

DESCRIBE 命令。你能够在 Web 站点 http://www.pearsonhighered.com/cs-resources 找到该作业的全部细节以及 RTSP 协议的概述。

人物专访


Henning Schulzrinne

Henning Schulzrinne 是哥伦比亚大学的教授、计算机科学系的主任和因特网实时实验室的负责人。他是 RTP、RTSP、SIP 和 GIST 这些音频和视频经因特网通信的关键协议的作者之一。Henning 在德国达姆施塔特工业大学 (TU Darmstadt) 获得了电气工业工程学士学位，在辛辛那提大学获得了电气和计算机工程硕士学位，以及在马萨诸塞大学阿默斯特学院获得了电气工程博士学位。

- 是什么使得您决定致力于多媒体网络？

这几乎是巧合。作为一个博士生，我从事 DARTnet 方面的工作，DARTnet 是一个用 T1 线路跨越美国的实验网络。DARTnet 用于为多播和因特网实时工具提供场所。这促使我写了我的第一个音频工具 NeVoT。通过一些 DARTnet 的参与者，我开始参与 IETF 和那时新成立的音频视频传输工作组的工作。这个工作组后来完成了 RTP 的标准化。

- 您在计算机行业中的第一份工作是什么？它给了您什么收获？

我在计算机行业的第一份工作是在加利福尼亚的 Livermore 读高中时焊接一个牵牛星（Altair）计算机工具包。回到德国，我开了一个小咨询公司来给旅行社设计一个地址管理程序，为我们的 TRS-80 开发了将数据存储在磁带上的功能，并通过一个自己制作的硬件接口把 IBM 的电动打字机作为打印机使用。

我第一份真正的工作是在 AT&T 的贝尔实验室，为在实验室环境下构建实验网络而研发的一个网络仿真器。

- 因特网实时实验室的目的是什么？

我们的目的是为因特网作为单一的未来通信基础设施提供组件和构件模块。这包括开发新协议，如 GIST（用于网络层信令）和 LoST（用于由位置寻找资源），或通过在丰富呈现、对等系统、下一代紧急情况呼叫和服务产生工具等方面的工作加强我们以前从事的协议，如 SIP。最近，我们也大规模地研究了用于 VoIP 的无线系统，因为 802.11b 和 802.11n 网络以及也许 WiMAX 网络有可能成为用于电话的重要的最后 1 英里技术。我们也试图使用一种称为 DYSWIS（Do You See What I See，你所见即我所见）的对等方到对等方故障诊断系统，大大改进用户在面对提供商和设备的复杂、混乱问题时的故障诊断能力。

通过构建原型和开放源码系统、测量实际系统的性能，以及对 IETF 标准做出贡献，我们试图做些实践性相关工作。

- 您对多媒体网络的未来展望是什么？

我们现在正处于一种过渡阶段，距离 IP 成为从 IPTV 到 VoIP 等多媒体服务的通用平台只有几年之遥了。我们期望着收音机、电话和电视即使在暴风雨和地震中都能工作，所以当因特网接替了这些专用网络的职责时，用户将期待有同样级别的可靠性。

我们将不得不学会为一个生态系统设计网络技术，该生态系统包括竞争的电信公司、服务和内容提供商，服务于大量技术上未受训练的用户，并保护他们免受少数但具有破坏性的一批恶意和犯罪用户的侵害。改变协议变得日益艰难。协议也变得更为复杂，因为它们需要考虑竞争的商业利益、安全性、隐私以及由防火墙和网络地址转换引起的网络透明性的欠缺。

因为多媒体网络正在成为几乎所有消费者娱乐的基础，因此以低成本管理非常大的网络将成为重点。用户将期待易于使用的网络，例如可以在所有的设备上找到相同的内容。

- 为什么 SIP 的未来很有前途？

随着现在的无线网络正在向 3G 网络升级，希望单个多媒体信令机制能够跨越所有类型的网络，包括从电缆调制解调器到企业电话网和公用无线网络。这连同软件无线电一道，在将来使下列东西成为可能：单一设备能被用于家庭网络中，能被作为无绳蓝牙电话，用于经 802.11 的企业网中和经 3G 网络的

广域网中。即使在我们有这样一个通用的无线设备之前，个人移动机制使得隐藏网络之间差别成为可能。一个标识成为找到一个人的通用方法，而不必记住或者分发一堆特定技术或特定位置的电话号码。

SIP 还将提供语音（比特）传输和语音服务分离。现在打破本地电话垄断在技术上已成为可能，即一个公司提供中性的比特传输，其他的公司提供 IP“拨号音”和常用的电话服务，例如网关、呼叫转移和主叫者 ID。

除了多媒体信令，SIP 提供在因特网中缺少的一种新服务：事件通知。我们已经有了这种具有 HTTP 不完善系统和电子邮件功能的近似服务，但是这决不会令人非常满意。因为事件是对分布式系统的通用抽象，这样可能简化新服务的构建。

- 您对进入网络领域的学生有什么忠告吗？

网络沟通了多个学科。它源于电子工程和计算机科学、运筹学、经济学和其他学科的所有方面。因此，网络研究者必须熟悉除了协议和路由选择算法以外的主题。

既然网络已经成为日常生活中如此重要的一部分，要在该领域标新立异的学生们应当思考网络中新的资源限制：人的时间及努力，而不只是带宽或存储。

从事网络研究工作能够给人以极大的满足，因为它使得人们能够相互通信和交换思想，这是人类所必需的。因特网服务提供商已经成为第三大全球性基础设施，接近于运输系统和能源分配。经济领域几乎没有哪个部分离开高性能网络还能正常运转，由此对可预测的将来应当有大量机会。

参考文献

有关 URL 的说明。在下面的引用中，我们提供了 Web 网页、仅在 Web 上有的文档和没有被会议或杂志出版的其他材料的 URL（当我们能够指出这些材料的位置时）。我们没有提供有关会议和杂志出版物的 URL，因为这些文档通常能够通过如下方式找到：使用某个搜索引擎，经该会议的 Web 站点（例如在所有 ACM SIGCOMM 会议和专题讨论会中的文章能够通过 http://www.acm.org/sigcomm 找到），或通过订阅数字图书馆。尽管到 2016 年 1 月，下面提供的所有 URL 都是有效的，但 URL 可能会过期。对于过期的文献，请参考本书的在线版本（http://www.pearsonhighered.com/cs-resources）。

有关因特网请求评论（RFC）的说明。因特网 RFC 的副本在多个网站上都可找到。因特网协会（监管 RFC 文档的组织）的 RFC 编辑们维护着网站 http://www.rfc-editor.org。该网站允许你通过标题、编号或作者来搜索某个特定的 RFC 文档，并将显示出对任何所列 RFC 的更新。因特网 RFC 可以被后面的 RFC 所更新或淘汰。我们喜欢的获取 RFC 文档的网站是初始 RFC 源，即 http://www.rfc-editor.org。

[3GPP 2016] Third Generation Partnership Project homepage, http://www.3gpp.org/

[Abramson 1970] N. Abramson, "The Aloha System—Another Alternative for Computer Communications," *Proc. 1970 Fall Joint Computer Conference, AFIPS Conference*, p. 37, 1970.

[Abramson 1985] N. Abramson, "Development of the Alohanet," *IEEE Transactions on Information Theory*, Vol. IT-31, No. 3 (Mar. 1985), pp. 119–123.

[Abramson 2009] N. Abramson, "The Alohanet—Surfing for Wireless Data," *IEEE Communications Magazine*, Vol. 47, No. 12, pp. 21–25.

[Adhikari 2011a] V. K. Adhikari, S. Jain, Y. Chen, Z. L. Zhang, "Vivisecting YouTube: An Active Measurement Study," Technical Report, University of Minnesota, 2011.

[Adhikari 2012] V. K. Adhikari, Y. Gao, F. Hao, M. Varvello, V. Hilt, M. Steiner, Z. L. Zhang, "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery," Technical Report, University of Minnesota, 2012.

[Afanasyev 2010] A. Afanasyev, N. Tilley, P. Reiher, L. Kleinrock, "Host-to-Host Congestion Control for TCP," *IEEE Communications Surveys & Tutorials*, Vol. 12, No. 3, pp. 304–342.

[Agarwal 2009] S. Agarwal, J. Lorch, "Matchmaking for Online Games and Other Latency-sensitive P2P Systems," *Proc. 2009 ACM SIGCOMM*.

[Ager 2012] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, W. Willinger, "Anatomy of a Large European ISP," Sigcomm, 2012.

[Ahn 1995] J. S. Ahn, P. B. Danzig, Z. Liu, and Y. Yan, "Experience with TCP Vegas: Emulation and Experiment," *Proc. 1995 ACM SIGCOMM* (Boston, MA, Aug. 1995), pp. 185–195.

[Akamai 2016] Akamai homepage, http://www.akamai.com

[Akella 2003] A. Akella, S. Seshan, A. Shaikh, "An Empirical Evaluation of Wide-Area Internet Bottlenecks," *Proc. 2003 ACM Internet Measurement Conference*

(Miami, FL, Nov. 2003).

[Akhshabi 2011] S. Akhshabi, A. C. Begen, C. Dovrolis, "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP," *Proc. 2011 ACM Multimedia Systems Conf.*

[Akyildiz 2010] I. Akyildiz, D. Gutierrex-Estevez, E. Reyes, "The Evolution to 4G Cellular Systems, LTE Advanced," *Physical Communication*, Elsevier, 3 (2010), 217–244.

[Albitz 1993] P. Albitz and C. Liu, *DNS and BIND*, O'Reilly & Associates, Petaluma, CA, 1993.

[Al-Fares 2008] M. Al-Fares, A. Loukissas, A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *Proc. 2008 ACM SIGCOMM*.

[Amazon 2014] J. Hamilton, *"AWS: Innovation at Scale*, YouTube video, https://www.youtube.com/watch?v=JIQETrFC_SQ

[Anderson 1995] J. B. Andersen, T. S. Rappaport, S. Yoshida, "Propagation Measurements and Models for Wireless Communications Channels," *IEEE Communications Magazine*, (Jan. 1995), pp. 42–49.

[Alizadeh 2010] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan. "Data center TCP (DCTCP)," *ACM SIGCOMM 2010 Conference*, ACM, New York, NY, USA, pp. 63–74.

[Allman 2011] E. Allman, "The Robustness Principle Reconsidered: Seeking a Middle Ground," *Communications of the ACM*, Vol. 54, No. 8 (Aug. 2011), pp. 40–45.

[Appenzeller 2004] G. Appenzeller, I. Keslassy, N. McKeown, "Sizing Router Buffers," *Proc. 2004 ACM SIGCOMM* (Portland, OR, Aug. 2004).

[ASO-ICANN 2016] The Address Supporting Organization homepage, http://www.aso.icann.org

[AT&T 2013] "AT&T Vision Alignment Challenge Technology Survey," AT&T Domain 2.0 Vision White Paper, November 13, 2013.

[Atheros 2016] Atheros Communications Inc., "Atheros AR5006 WLAN Chipset Product Bulletins," http://www.atheros.com/pt/AR5006Bulletins.htm

[Ayanoglu 1995] E. Ayanoglu, S. Paul, T. F. La Porta, K. K. Sabnani, R. D. Gitlin, "AIRMAIL: A Link-Layer Protocol for Wireless Networks," *ACM ACM/Baltzer Wireless Networks Journal*, 1: 47–60, Feb. 1995.

[Bakre 1995] A. Bakre, B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. 1995 Int. Conf. on Distributed Computing Systems (ICDCS)* (May 1995), pp. 136–143.

[Balakrishnan 1997] H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz, "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links," *IEEE/ACM Transactions on Networking* Vol. 5, No. 6 (Dec. 1997).

[Balakrishnan 2003] H. Balakrishnan, F. Kaashoek, D. Karger, R. Morris, I. Stoica, "Looking Up Data in P2P Systems," *Communications of the ACM*, Vol. 46, No. 2 (Feb. 2003), pp. 43–48.

[Baldauf 2007] M. Baldauf, S. Dustdar, F. Rosenberg, "A Survey on Context-Aware Systems," *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 2, No. 4 (2007), pp. 263–277.

[Baran 1964] P. Baran, "On Distributed Communication Networks," *IEEE Transactions on Communication Systems*, Mar. 1964. Rand Corporation Technical report

with the same title (Memorandum RM-3420-PR, 1964). http://www.rand.org/publications/RM/RM3420/

[Bardwell 2004] J. Bardwell, "You Believe You Understand What You Think I Said . . . The Truth About 802.11 Signal and Noise Metrics: A Discussion Clarifying Often-Misused 802.11 WLAN Terminologies," http://www.connect802.com/download/techpubs/2004/you_believe_D100201.pdf

[Barford 2009] P. Barford, N. Duffield, A. Ron, J. Sommers, "Network Performance Anomaly Detection and Localization," *Proc. 2009 IEEE INFOCOM* (Apr. 2009).

[Baronti 2007] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, Y. Hu, "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards," *Computer Communications*, Vol. 30, No. 7 (2007), pp. 1655–1695.

[Baset 2006] S. A. Basset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," *Proc. 2006 IEEE INFOCOM* (Barcelona, Spain, Apr. 2006).

[BBC 2001] BBC news online "A Small Slice of Design," Apr. 2001, http://news.bbc.co.uk/2/hi/science/nature/1264205.stm

[Beheshti 2008] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, G. Salmon, "Experimental Study of Router Buffer Sizing," *Proc. ACM Internet Measurement Conference* (Oct. 2008, Vouliagmeni, Greece).

[Bender 2000] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, A. Viterbi, "CDMA/HDR: A Bandwidth-Efficient High-Speed Wireless Data Service for Nomadic Users," *IEEE Commun. Mag.*, Vol. 38, No. 7 (July 2000), pp. 70–77.

[Berners-Lee 1989] T. Berners-Lee, CERN, "Information Management: A Proposal," Mar. 1989, May 1990. http://www.w3.org/History/1989/proposal.html

[Berners-Lee 1994] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, A. Secret, "The World-Wide Web," *Communications of the ACM*, Vol. 37, No. 8 (Aug. 1994), pp. 76–82.

[Bertsekas 1991] D. Bertsekas, R. Gallagher, *Data Networks, 2nd Ed.*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[Biersack 1992] E. W. Biersack, "Performance Evaluation of Forward Error Correction in ATM Networks," *Proc. 1999 ACM SIGCOMM* (Baltimore, MD, Aug. 1992), pp. 248–257.

[BIND 2016] Internet Software Consortium page on BIND, http://www.isc.org/bind.html

[Bisdikian 2001] C. Bisdikian, "An Overview of the Bluetooth Wireless Technology," *IEEE Communications Magazine*, No. 12 (Dec. 2001), pp. 86–94.

[Bishop 2003] M. Bishop, *Computer Security: Art and Science*, Boston: Addison Wesley, Boston MA, 2003.

[Black 1995] U. Black, *ATM Volume I: Foundation for Broadband Networks*, Prentice Hall, 1995.

[Black 1997] U. Black, *ATM Volume II: Signaling in Broadband Networks*, Prentice Hall, 1997.

[Blumenthal 2001] M. Blumenthal, D. Clark, "Rethinking the Design of the

Internet: The End-to-end Arguments vs. the Brave New World," *ACM Transactions on Internet Technology*, Vol. 1, No. 1 (Aug. 2001), pp. 70–109.

[Bochman 1984] G. V. Bochmann, C. A. Sunshine, "Formal Methods in Communication Protocol Design," *IEEE Transactions on Communications*, Vol. 28, No. 4 (Apr. 1980) pp. 624–631.

[Bolot 1996] J-C. Bolot, A. Vega-Garcia, "Control Mechanisms for Packet Audio in the Internet," *Proc. 1996 IEEE INFOCOM*, pp. 232–239.

[Bosshart 2013] P. Bosshart, G. Gibb, H. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, M. Horowitz, "Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN," *ACM SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 99–110.

[Bosshart 2014] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, "P4: Programming Protocol-Independent Packet Processors," *ACM SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), pp. 87–95.

[Brakmo 1995] L. Brakmo, L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal of Selected Areas in Communications*, Vol. 13, No. 8 (Oct. 1995), pp. 1465–1480.

[Bryant 1988] B. Bryant, "Designing an Authentication System: A Dialogue in Four Scenes," http://web.mit.edu/kerberos/www/dialogue.html

[Bush 1945] V. Bush, "As We May Think," *The Atlantic Monthly*, July 1945. http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

[Byers 1998] J. Byers, M. Luby, M. Mitzenmacher, A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," *Proc. 1998 ACM SIGCOMM* (Vancouver, Canada, Aug. 1998), pp. 56–67.

[Caesar 2005a] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, J. van der Merwe, "Design and implementation of a Routing Control Platform," *Proc. Networked Systems Design and Implementation* (May 2005).

[Caesar 2005b] M. Caesar, J. Rexford, "BGP Routing Policies in ISP Networks," *IEEE Network Magazine*, Vol. 19, No. 6 (Nov. 2005).

[Caldwell 2012] C. Caldwell, "The Prime Pages," http://www.utm.edu/research/primes/prove

[Cardwell 2000] N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency," *Proc. 2000 IEEE INFOCOM* (Tel-Aviv, Israel, Mar. 2000).

[Casado 2007] M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, "Ethane: Taking Control of the Enterprise," *Proc. ACM SIGCOMM '07*, New York, pp. 1–12. See also *IEEE/ACM Trans. Networking*, 17, 4 (Aug. 2007), pp. 270–1283.

[Casado 2009] M. Casado, M. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, S. Shenker, "Rethinking Enterprise Network Control," *IEEE/ACM Transactions on Networking (ToN)*, Vol. 17, No. 4 (Aug. 2009), pp. 1270–1283.

[Casado 2014] M. Casado, N. Foster, A. Guha, "Abstractions for Software-Defined Networks," *Communications of the ACM*, Vol. 57 No. 10, (Oct. 2014), pp. 86–95.

[Cerf 1974] V. Cerf, R. Kahn, "A Protocol for Packet Network Interconnection," *IEEE Transactions on Communications Technology*, Vol. COM-22, No. 5, pp. 627–641.

[CERT 2001–09] CERT, "Advisory 2001–09: Statistical Weaknesses in TCP/IP

Initial Sequence Numbers," http://www.cert.org/advisories/CA-2001-09.html

[CERT 2003–04] CERT, "CERT Advisory CA-2003-04 MS-SQL Server Worm," http://www.cert.org/advisories/CA-2003-04.html

[CERT 2016] CERT, http://www.cert.org

[CERT Filtering 2012] CERT, "Packet Filtering for Firewall Systems," http://www.cert.org/tech_tips/packet_filtering.html

[Cert SYN 1996] CERT, "Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks," http://www.cert.org/advisories/CA-1998-01.html

[Chandra 2007] T. Chandra, R. Greisemer, J. Redstone, "Paxos Made Live: an Engineering Perspective," Proc. of 2007 ACM Symposium on Principles of Distributed Computing (PODC), pp. 398–407.

[Chao 2001] H. J. Chao, C. Lam, E. Oki, *Broadband Packet Switching Technologies—A Practical Guide to ATM Switches and IP Routers*, John Wiley & Sons, 2001.

[Chao 2011] C. Zhang, P. Dunghel, D. Wu, K. W. Ross, "Unraveling the BitTorrent Ecosystem," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 7 (July 2011).

[Chen 2000] G. Chen, D. Kotz, "A Survey of Context-Aware Mobile Computing Research," *Technical Report TR2000-381*, Dept. of Computer Science, Dartmouth College, Nov. 2000. http://www.cs.dartmouth.edu/reports/TR2000-381.pdf

[Chen 2006] K.-T. Chen, C.-Y. Huang, P. Huang, C.-L. Lei, "Quantifying Skype User Satisfaction," *Proc. 2006 ACM SIGCOMM* (Pisa, Italy, Sept. 2006).

[Chen 2011] Y. Chen, S. Jain, V. K. Adhikari, Z. Zhang, "Characterizing Roles of Front-End Servers in End-to-End Performance of Dynamic Content Distribution," *Proc. 2011 ACM Internet Measurement Conference* (Berlin, Germany, Nov. 2011).

[Cheswick 2000] B. Cheswick, H. Burch, S. Branigan, "Mapping and Visualizing the Internet," *Proc. 2000 Usenix Conference* (San Diego, CA, June 2000).

[Chiu 1989] D. Chiu, R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems*, Vol. 17, No. 1, pp. 1–14. http://www.cs.wustl.edu/~jain/papers/cong_av.htm

[Christiansen 2001] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, "Tuning Red for Web Traffic," *IEEE/ACM Transactions on Networking*, Vol. 9, No. 3 (June 2001), pp. 249–264.

[Chuang 2005] S. Chuang, S. Iyer, N. McKeown, "Practical Algorithms for Performance Guarantees in Buffered Crossbars," *Proc. 2005 IEEE INFOCOM.*

[Cisco 802.11ac 2014] Cisco Systems, "802.11ac: The Fifth Generation of Wi-Fi," Technical White Paper, Mar. 2014.

[Cisco 7600 2016] Cisco Systems, "Cisco 7600 Series Solution and Design Guide," http://www.cisco.com/en/US/products/hw/routers/ps368/prod_technical_reference09186a0080092246.html

[Cisco 8500 2012] Cisco Systems Inc., "Catalyst 8500 Campus Switch Router Architecture," http://www.cisco.com/univercd/cc/td/doc/product/l3sw/8540/rel_12_0/w5_6f/softcnfg/1cfg8500.pdf

[Cisco 12000 2016] Cisco Systems Inc., "Cisco XR 12000 Series and Cisco 12000 Series Routers," http://www.cisco.com/en/US/products/ps6342/index.html

[Cisco 2012] Cisco 2012, Data Centers, http://www.cisco.com/go/dce

[Cisco 2015] Cisco Visual Networking Index: Forecast and Methodology, 2014–2019, White Paper, 2015.

[Cisco 6500 2016] Cisco Systems, "Cisco Catalyst 6500 Architecture White Paper," http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/prod_white_paper0900aecd80673385.html

[Cisco NAT 2016] Cisco Systems Inc., "How NAT Works," http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094831.shtml

[Cisco QoS 2016] Cisco Systems Inc., "Advanced QoS Services for the Intelligent Internet," http://www.cisco.com/warp/public/cc/pd/iosw/ioft/ioqo/tech/qos_wp.htm

[Cisco Queue 2016] Cisco Systems Inc., "Congestion Management Overview," http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfconmg.html

[Cisco SYN 2016] Cisco Systems Inc., "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks," http://www.cisco.com/en/US/tech/tk828/technologies_tech_note09186a00800f67d5.shtml

[Cisco TCAM 2014] Cisco Systems Inc., "CAT 6500 and 7600 Series Routers and Switches TCAM Allocation Adjustment Procedures," http://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/117712-problemsolution-cat6500-00.html

[Cisco VNI 2015] Cisco Systems Inc., "Visual Networking Index," http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html

[Clark 1988] D. Clark, "The Design Philosophy of the DARPA Internet Protocols," *Proc. 1988 ACM SIGCOMM* (Stanford, CA, Aug. 1988).

[Cohen 1977] D. Cohen, "Issues in Transnet Packetized Voice Communication," *Proc. Fifth Data Communications Symposium* (Snowbird, UT, Sept. 1977), pp. 6–13.

[Cookie Central 2016] Cookie Central homepage, http://www.cookiecentral.com/n_cookie_faq.htm

[Cormen 2001] T. H. Cormen, *Introduction to Algorithms*, 2nd Ed., MIT Press, Cambridge, MA, 2001.

[Crow 1997] B. Crow, I. Widjaja, J. Kim, P. Sakai, "IEEE 802.11 Wireless Local Area Networks," *IEEE Communications Magazine* (Sept. 1997), pp. 116–126.

[Cusumano 1998] M. A. Cusumano, D. B. Yoffie, *Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft*, Free Press, New York, NY, 1998.

[Czyz 2014] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, M. Bailey, "Measuring IPv6 Adoption," *Proc. ACM SIGCOMM 2014*, ACM, New York, NY, USA, pp. 87–98.

[Dahlman 1998] E. Dahlman, B. Gudmundson, M. Nilsson, J. Sköld, "UMTS/IMT-2000 Based on Wideband CDMA," *IEEE Communications Magazine* (Sept. 1998), pp. 70–80.

[Daigle 1991] J. N. Daigle, *Queuing Theory for Telecommunications*, Addison-Wesley, Reading, MA, 1991.

[DAM 2016] Digital Attack Map, http://www.digitalattackmap.com

[Davie 2000] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*, Morgan Kaufmann Series in Networking, 2000.

[Davies 2005] G. Davies, F. Kelly, "Network Dimensioning, Service Costing, and Pricing in a Packet-Switched Environment," *Telecommunications Policy*, Vol. 28, No. 4, pp. 391–412.

[DEC 1990] Digital Equipment Corporation, "In Memoriam: J. C. R. Licklider 1915–1990," SRC Research Report 61, Aug. 1990. http://www.memex.org/licklider.pdf

[DeClercq 2002] J. DeClercq, O. Paridaens, "Scalability Implications of Virtual Private Networks," *IEEE Communications Magazine*, Vol. 40, No. 5 (May 2002), pp. 151–157.

[Demers 1990] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *Internetworking: Research and Experience*, Vol. 1, No. 1 (1990), pp. 3–26.

[dhc 2016] IETF Dynamic Host Configuration working group homepage, http://www.ietf.org/html.charters/dhc-charter.html

[Dhungel 2012] P. Dhungel, K. W. Ross, M. Steiner., Y. Tian, X. Hei, "Xunlei: Peer-Assisted Download Acceleration on a Massive Scale," *Passive and Active Measurement Conference (PAM) 2012*, Vienna, 2012.

[Diffie 1976] W. Diffie, M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, Vol IT-22 (1976), pp. 644–654.

[Diggavi 2004] S. N. Diggavi, N. Al-Dhahir, A. Stamoulis, R. Calderbank, "Great Expectations: The Value of Spatial Diversity in Wireless Networks," *Proceedings of the IEEE*, Vol. 92, No. 2 (Feb. 2004).

[Dilley 2002] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, B. Weihl, "Globally Distributed Content Delivert," *IEEE Internet Computing* (Sept.–Oct. 2002).

[Diot 2000] C. Diot, B. N. Levine, B. Lyles, H. Kassem, D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network*, Vol. 14, No. 1 (Jan./Feb. 2000) pp. 78–88.

[Dischinger 2007] M. Dischinger, A. Haeberlen, K. Gummadi, S. Saroiu, "Characterizing residential broadband networks," *Proc. 2007 ACM Internet Measurement Conference*, pp. 24–26.

[Dmitiropoulos 2007] X. Dmitiropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. C. Claffy, G. Riley, "AS Relationships: Inference and Validation," *ACM Computer Communication Review* (Jan. 2007).

[DOCSIS 2011] Data-Over-Cable Service Interface Specifications, DOCSIS 3.0: MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.0-I16-110623, 2011.

[Dodge 2016] M. Dodge, "An Atlas of Cyberspaces," http://www.cybergeography.org/atlas/isp_maps.html

[Donahoo 2001] M. Donahoo, K. Calvert, *TCP/IP Sockets in C: Practical Guide for Programmers*, Morgan Kaufman, 2001.

[DSL 2016] DSL Forum homepage, http://www.dslforum.org/

[Dhunghel 2008] P. Dhungel, D. Wu, B. Schonhorst, K.W. Ross, "A Measurement Study of Attacks on BitTorrent Leechers," *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)* (Tampa Bay, FL, Feb. 2008).

[Droms 2002] R. Droms, T. Lemon, *The DHCP Handbook* (2nd Edition), SAMS Publishing, 2002.

[Edney 2003] J. Edney and W. A. Arbaugh, *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*, Addison-Wesley Professional, 2003.

[Edwards 2011] W. K. Edwards, R. Grinter, R. Mahajan, D. Wetherall, "Advancing the State of Home Networking," *Communications of the ACM*, Vol. 54, No. 6 (June 2011), pp. 62–71.

[Ellis 1987] H. Ellis, "The Story of Non-Secret Encryption," http://jya.com/ellis-doc.htm

[Erickson 2013] D. Erickson, " The Beacon Openflow Controller," 2nd *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (HotSDN '13). ACM, New York, NY, USA, pp. 13–18.

[Ericsson 2012] Ericsson, "The Evolution of Edge," http://www.ericsson.com/technology/whitepapers/broadband/evolution_of_EDGE.shtml

[Facebook 2014] A. Andreyev, "Introducing Data Center Fabric, the Next-Generation Facebook Data Center Network," https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network

[Faloutsos 1999] C. Faloutsos, M. Faloutsos, P. Faloutsos, "What Does the Internet Look Like? Empirical Laws of the Internet Topology," *Proc. 1999 ACM SIGCOMM* (Boston, MA, Aug. 1999).

[Farrington 2010] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers," *Proc. 2010 ACM SIGCOMM*.

[Feamster 2004] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, K. van der Merwe, "The Case for Separating Routing from Routers," *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Sept. 2004.

[Feamster 2004] N. Feamster, J. Winick, J. Rexford, "A Model for BGP Routing for Network Engineering," *Proc. 2004 ACM SIGMETRICS* (New York, NY, June 2004).

[Feamster 2005] N. Feamster, H. Balakrishnan, "Detecting BGP Configuration Faults with Static Analysis," *NSDI* (May 2005).

[Feamster 2013] N. Feamster, J. Rexford, E. Zegura, "The Road to SDN," *ACM Queue*, Volume 11, Issue 12, (Dec. 2013).

[Feldmeier 1995] D. Feldmeier, "Fast Software Implementation of Error Detection Codes," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 6 (Dec. 1995), pp. 640–652.

[Ferguson 2013] A. Ferguson, A. Guha, C. Liang, R. Fonseca, S. Krishnamurthi, "Participatory Networking: An API for Application Control of SDNs," *Proceedings ACM SIGCOMM 2013*, pp. 327–338.

[Fielding 2000] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000. PhD Thesis, UC Irvine, 2000.

[FIPS 1995] Federal Information Processing Standard, "Secure Hash Standard," FIPS Publication 180-1. http://www.itl.nist.gov/fipspubs/fip180-1.htm

[Floyd 1999] S. Floyd, K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 5 (Oct. 1998), pp. 458–472.

[Floyd 2000] S. Floyd, M. Handley, J. Padhye, J. Widmer, "Equation-Based Congestion Control for Unicast Applications," *Proc. 2000 ACM SIGCOMM* (Stockholm, Sweden, Aug. 2000).

[Floyd 2001] S. Floyd, "A Report on Some Recent Developments in TCP Congestion Control," *IEEE Communications Magazine* (Apr. 2001).

[Floyd 2016] S. Floyd, "References on RED (Random Early Detection) Queue Management," http://www.icir.org/floyd/red.html

[Floyd Synchronization 1994] S. Floyd, V. Jacobson, "Synchronization of Periodic Routing Messages," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2 (Apr. 1997) pp. 122–136.

[Floyd TCP 1994] S. Floyd, "TCP and Explicit Congestion Notification," *ACM SIGCOMM Computer Communications Review*, Vol. 24, No. 5 (Oct. 1994), pp. 10–23.

[Fluhrer 2001] S. Fluhrer, I. Mantin, A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," *Eighth Annual Workshop on Selected Areas in Cryptography* (Toronto, Canada, Aug. 2002).

[Fortz 2000] B. Fortz, M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," *Proc. 2000 IEEE INFOCOM* (Tel Aviv, Israel, Apr. 2000).

[Fortz 2002] B. Fortz, J. Rexford, M. Thorup, "Traffic Engineering with Traditional IP Routing Protocols," *IEEE Communication Magazine* (Oct. 2002).

[Fraleigh 2003] C. Fraleigh, F. Tobagi, C. Diot, "Provisioning IP Backbone Networks to Support Latency Sensitive Traffic," *Proc. 2003 IEEE INFOCOM* (San Francisco, CA, Mar. 2003).

[Frost 1994] J. Frost, "BSD Sockets: A Quick and Dirty Primer," http://world.std.com/~jimf/papers/sockets/sockets.html

[FTC 2015] Internet of Things: Privacy and Security in a Connected World, Federal Trade Commission, 2015, https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf

[FTTH 2016] Fiber to the Home Council, http://www.ftthcouncil.org/

[Gao 2001] L. Gao, J. Rexford, "Stable Internet Routing Without Global Coordination," *IEEE/ACM Transactions on Networking*, Vol. 9, No. 6 (Dec. 2001), pp. 681–692.

[Gartner 2014] Gartner report on Internet of Things, http://www.gartner.com/technology/research/internet-of-things

[Gauthier 1999] L. Gauthier, C. Diot, and J. Kurose, "End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet," *Proc. 1999 IEEE INFOCOM* (New York, NY, Apr. 1999).

[Gember-Jacobson 2014] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, A. Akella, "OpenNF: Enabling Innovation in Network Function Control," *Proc. ACM SIGCOMM 2014*, pp. 163–174.

[Goodman 1997] David J. Goodman, *Wireless Personal Communications Systems*, Prentice-Hall, 1997.

[Google IPv6 2015] Google Inc. "IPv6 Statistics," https://www.google.com/intl/en/ipv6/statistics.html

[Google Locations 2016] Google data centers. http://www.google.com/corporate/datacenter/locations.html

[Goralski 1999] W. Goralski, *Frame Relay for High-Speed Networks*, John Wiley, New York, 1999.

[Greenberg 2009a] A. Greenberg, J. Hamilton, D. Maltz, P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *ACM Computer Communications Review* (Jan. 2009).

[Greenberg 2009b] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," *Proc. 2009 ACM SIGCOMM*.

[Greenberg 2011] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," *Communications of the ACM*, Vol. 54, No. 3 (Mar. 2011), pp. 95–104.

[Greenberg 2015] A. Greenberg, "SDN for the Cloud," Sigcomm 2015 Keynote Address, http://conferences.sigcomm.org/sigcomm/2015/pdf/papers/keynote.pdf

[Griffin 2012] T. Griffin, "Interdomain Routing Links," http://www.cl.cam.ac.uk/~tgg22/interdomain/

[Gude 2008] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks," *ACM SIGCOMM Computer Communication Review*, July 2008.

[Guha 2006] S. Guha, N. Daswani, R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," *Proc. Fifth Int. Workshop on P2P Systems* (Santa Barbara, CA, 2006).

[Guo 2005] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, "Measurement, Analysis, and Modeling of BitTorrent-Like Systems," *Proc. 2005 ACM Internet Measurement Conference*.

[Guo 2009] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," *Proc. 2009 ACM SIGCOMM*.

[Gupta 2001] P. Gupta, N. McKeown, "Algorithms for Packet Classification," *IEEE Network Magazine*, Vol. 15, No. 2 (Mar./Apr. 2001), pp. 24–32.

[Gupta 2014] A. Gupta, L. Vanbever, M. Shahbaz, S. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, E. Katz-Bassett, "SDX: A Software Defined Internet Exchange, " *Proc. ACM SIGCOMM 2014* (Aug. 2014), pp. 551–562.

[Ha 2008] S. Ha, I. Rhee, L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating System Review*, 2008.

[Halabi 2000] S. Halabi, *Internet Routing Architectures*, 2nd Ed., Cisco Press, 2000.

[Hanabali 2005] A. A. Hanbali, E. Altman, P. Nain, "A Survey of TCP over Ad Hoc Networks," *IEEE Commun. Surveys and Tutorials*, Vol. 7, No. 3 (2005), pp. 22–36.

[Hei 2007] X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, "A Measurement Study of a Large-scale P2P IPTV System," *IEEE Trans. on Multimedia* (Dec. 2007).

[Heidemann 1997] J. Heidemann, K. Obraczka, J. Touch, "Modeling the Performance of HTTP over Several Transport Protocols," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5 (Oct. 1997), pp. 616–630.

[Held 2001] G. Held, *Data Over Wireless Networks: Bluetooth, WAP, and Wireless LANs*, McGraw-Hill, 2001.

[Holland 2001] G. Holland, N. Vaidya, V. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," *Proc. 2001 ACM Int. Conference of Mobile Computing and Networking (Mobicom01)* (Rome, Italy, July 2001).

[Hollot 2002] C.V. Hollot, V. Misra, D. Towsley, W. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows," *IEEE Transactions on Automatic Control*, Vol. 47, No. 6 (June 2002), pp. 945–959.

[Hong 2013] C. Hong, S, Kandula, R. Mahajan, M.Zhang, V. Gill, M. Nanduri, R. Wattenhofer, "Achieving High Utilization with Software-driven WAN," ACM *SIGCOMM Conference* (Aug. 2013), pp.15–26.

[Huang 2002] C. Haung, V. Sharma, K. Owens, V. Makam, "Building Reliable MPLS Networks Using a Path Protection Mechanism," *IEEE Communications Magazine*, Vol. 40, No. 3 (Mar. 2002), pp. 156–162.

[Huang 2005] Y. Huang, R. Guerin, "Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger?," *Proc. IEEE Int. Conf. Network Protocols (ICNP)* (Boston MA, Nov. 2005).

[Huang 2008] C. Huang, J. Li, A. Wang, K. W. Ross, "Understanding Hybrid CDN-P2P: Why Limelight Needs Its Own Red Swoosh," *Proc. 2008 NOSSDAV*, Braunsch-

weig, Germany.

[Huitema 1998] C. Huitema, *IPv6: The New Internet Protocol*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1998.

[Huston 1999a] G. Huston, "Interconnection, Peering, and Settlements—Part I," *The Internet Protocol Journal*, Vol. 2, No. 1 (Mar. 1999).

[Huston 2004] G. Huston, "NAT Anatomy: A Look Inside Network Address Translators," *The Internet Protocol Journal*, Vol. 7, No. 3 (Sept. 2004).

[Huston 2008a] G. Huston, "Confronting IPv4 Address Exhaustion," http://www.potaroo.net/ispcol/2008-10/v4depletion.html

[Huston 2008b] G. Huston, G. Michaelson, "IPv6 Deployment: Just where are we?" http://www.potaroo.net/ispcol/2008-04/ipv6.html

[Huston 2011a] G. Huston, "A Rough Guide to Address Exhaustion," *The Internet Protocol Journal*, Vol. 14, No. 1 (Mar. 2011).

[Huston 2011b] G. Huston, "Transitioning Protocols," *The Internet Protocol Journal*, Vol. 14, No. 1 (Mar. 2011).

[IAB 2016] Internet Architecture Board homepage, http://www.iab.org/

[IANA Protocol Numbers 2016] Internet Assigned Numbers Authority, Protocol Numbers, http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

[IBM 1997] IBM Corp., *IBM Inside APPN - The Essential Guide to the Next-Generation SNA*, SG24-3669-03, June 1997.

[ICANN 2016] The Internet Corporation for Assigned Names and Numbers homepage, http://www.icann.org

[IEEE 802 2016] IEEE 802 LAN/MAN Standards Committee homepage, http://www.ieee802.org/

[IEEE 802.11 1999] IEEE 802.11, "1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Network—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[IEEE 802.11ac 2013] IEEE, "802.11ac-2013—IEEE Standard for Information technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Amendment 4: Enhancements for Very High Throughput for Operation in Bands Below 6 GHz."

[IEEE 802.11n 2012] IEEE, "IEEE P802.11—Task Group N—Meeting Update: Status of 802.11n," http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm

[IEEE 802.15 2012] IEEE 802.15 Working Group for WPAN homepage, http://grouper.ieee.org/groups/802/15/.

[IEEE 802.15.4 2012] IEEE 802.15 WPAN Task Group 4, http://www.ieee802.org/15/pub/TG4.html

[IEEE 802.16d 2004] IEEE, "IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems," http://standards.ieee.org/getieee802/download/802.16-2004.pdf

[IEEE 802.16e 2005] IEEE, "IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1," http://standards.ieee.org/getieee802/download/802.16e-2005.pdf

[IEEE 802.1q 2005] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks," http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf

[IEEE 802.1X] IEEE Std 802.1X-2001 Port-Based Network Access Control, http://standards.ieee.org/reading/ieee/std_public/description/lanman/802.1x-2001_desc.html

[IEEE 802.3 2012] IEEE, "IEEE 802.3 CSMA/CD (Ethernet)," http://grouper.ieee.org/groups/802/3/

[IEEE 802.5 2012] IEEE, IEEE 802.5 homepage, http://www.ieee802.org/5/www8025org/

[IETF 2016] Internet Engineering Task Force homepage, http://www.ietf.org

[Ihm 2011] S. Ihm, V. S. Pai, "Towards Understanding Modern Web Traffic," *Proc. 2011 ACM Internet Measurement Conference* (Berlin).

[IMAP 2012] The IMAP Connection, http://www.imap.org/

[Intel 2016] Intel Corp., "Intel 710 Ethernet Adapter," http://www.intel.com/content/www/us/en/ethernet-products/converged-network-adapters/ethernet-xl710.html

[Internet2 Multicast 2012] Internet2 Multicast Working Group homepage, http://www.internet2.edu/multicast/

[ISC 2016] Internet Systems Consortium homepage, http://www.isc.org

[ISI 1979] Information Sciences Institute, "DoD Standard Internet Protocol," Internet Engineering Note 123 (Dec. 1979), http://www.isi.edu/in-notes/ien/ien123.txt

[ISO 2016] International Organization for Standardization homepage, International Organization for Standardization, http://www.iso.org/

[ISO X.680 2002] International Organization for Standardization, "X.680: ITU-T Recommendation X.680 (2002) Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation," http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf

[ITU 1999] Asymmetric Digital Subscriber Line (ADSL) Transceivers. ITU-T G.992.1, 1999.

[ITU 2003] Asymmetric Digital Subscriber Line (ADSL) Transceivers—Extended Bandwidth ADSL2 (ADSL2Plus). ITU-T G.992.5, 2003.

[ITU 2005a] International Telecommunication Union, "ITU-T X.509, The Directory: Public-key and attribute certificate frameworks" (Aug. 2005).

[ITU 2006] ITU, "G.993.1: Very High Speed Digital Subscriber Line Transceivers (VDSL)," https://www.itu.int/rec/T-REC-G.993.1-200406-I/en, 2006.

[ITU 2015] "Measuring the Information Society Report," 2015, http://www.itu.int/en/ITU-D/Statistics/Pages/publications/mis2015.aspx

[ITU 2012] The ITU homepage, http://www.itu.int/

[ITU-T Q.2931 1995] International Telecommunication Union, "Recommendation

Q.2931 (02/95)—Broadband Integrated Services Digital Network (B-ISDN)—Digital Subscriber Signalling System No. 2 (DSS 2)—User-Network Interface (UNI)—Layer 3 Specification for Basic Call/Connection Control."

[IXP List 2016] List of IXPs, Wikipedia, https://en.wikipedia.org/wiki/List_of_Internet_exchange_points

[Iyengar 2015] J. Iyengar, I. Swett, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2," Internet Draft draft-tsvwg-quic-protocol-00, June 2015.

[Iyer 2008] S. Iyer, R. R. Kompella, N. McKeown, "Designing Packet Buffers for Router Line Cards," *IEEE Transactions on Networking*, Vol. 16, No. 3 (June 2008), pp. 705–717.

[Jacobson 1988] V. Jacobson, "Congestion Avoidance and Control," *Proc. 1988 ACM SIGCOMM* (Stanford, CA, Aug. 1988), pp. 314–329.

[Jain 1986] R. Jain, "A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks," *IEEE Journal on Selected Areas in Communications SAC-4*, 7 (Oct. 1986).

[Jain 1989] R. Jain, "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks," *ACM SIGCOMM Computer Communications Review*, Vol. 19, No. 5 (1989), pp. 56–71.

[Jain 1994] R. Jain, *FDDI Handbook: High-Speed Networking Using Fiber and Other Media*, Addison-Wesley, Reading, MA, 1994.

[Jain 1996] R. Jain. S. Kalyanaraman, S. Fahmy, R. Goyal, S. Kim, "Tutorial Paper on ABR Source Behavior," *ATM Forum*/96-1270, Oct. 1996. http://www.cse.wustl.edu/~jain/atmf/ftp/atm96-1270.pdf

[Jain 2013] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S.Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, A, Vahdat, "B4: Experience with a Globally Deployed Software Defined Wan," *ACM SIGCOMM 2013*, pp. 3–14.

[Jaiswal 2003] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP backbone," *Proc. 2003 IEEE INFOCOM*.

[Ji 2003] P. Ji, Z. Ge, J. Kurose, D. Towsley, "A Comparison of Hard-State and Soft-State Signaling Protocols," *Proc. 2003 ACM SIGCOMM* (Karlsruhe, Germany, Aug. 2003).

[Jimenez 1997] D. Jimenez, "Outside Hackers Infiltrate MIT Network, Compromise Security," *The Tech*, Vol. 117, No 49 (Oct. 1997), p. 1, http://www-tech.mit.edu/V117/N49/hackers.49n.html

[Jin 2004] C. Jin, D. X. We, S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *Proc. 2004 IEEE INFOCOM* (Hong Kong, Mar. 2004).

[Juniper Contrail 2016] Juniper Networks, "Contrail," http://www.juniper.net/us/en/products-services/sdn/contrail/

[Juniper MX2020 2015] Juniper Networks, "MX2020 and MX2010 3D Universal Edge Routers," www.juniper.net/us/en/local/pdf/.../1000417-en.pdf

[Kaaranen 2001] H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen, V. Niemi, *Networks: Architecture, Mobility and Services*, New York: John Wiley & Sons, 2001.

[Kahn 1967] D. Kahn, *The Codebreakers: The Story of Secret Writing*, The Macmillan Company, 1967.

[Kahn 1978] R. E. Kahn, S. Gronemeyer, J. Burchfiel, R. Kunzelman, "Advances in Packet Radio Technology," *Proc. 1978 IEEE INFOCOM*, 66, 11 (Nov. 1978).

[Kamerman 1997] A. Kamerman, L. Monteban, "WaveLAN-II: A High–Performance Wireless LAN for the Unlicensed Band," *Bell Labs Technical Journal* (Summer 1997), pp. 118–133.

[Kar 2000] K. Kar, M. Kodialam, T. V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *IEEE J. Selected Areas in Communications* (Dec. 2000).

[Karn 1987] P. Karn, C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," *Proc. 1987 ACM SIGCOMM*.

[Karol 1987] M. Karol, M. Hluchyj, A. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, Vol. 35, No. 12 (Dec.1987), pp. 1347–1356.

[Kaufman 1995] C. Kaufman, R. Perlman, M. Speciner, *Network Security, Private Communication in a Public World*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[Kelly 1998] F. P. Kelly, A. Maulloo, D. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *J. Operations Res. Soc.*, Vol. 49, No. 3 (Mar. 1998), pp. 237–252.

[Kelly 2003] T. Kelly, "Scalable TCP: Improving Performance in High Speed Wide Area
Networks," *ACM SIGCOMM Computer Communications Review*, Volume 33, No. 2 (Apr. 2003), pp 83–91.

[Kilkki 1999] K. Kilkki, *Differentiated Services for the Internet*, Macmillan Technical Publishing, Indianapolis, IN, 1999.

[Kim 2005] H. Kim, S. Rixner, V. Pai, "Network Interface Data Caching," *IEEE Transactions on Computers*, Vol. 54, No. 11 (Nov. 2005), pp. 1394–1408.

[Kim 2008] C. Kim, M. Caesar, J. Rexford, "Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises," *Proc. 2008 ACM SIGCOMM* (Seattle, WA, Aug. 2008).

[Kleinrock 1961] L. Kleinrock, "Information Flow in Large Communication Networks," RLE Quarterly Progress Report, July 1961.

[Kleinrock 1964] L. Kleinrock, *1964 Communication Nets: Stochastic Message Flow and Delay*, McGraw-Hill, New York, NY, 1964.

[Kleinrock 1975] L. Kleinrock, *Queuing Systems, Vol. 1*, John Wiley, New York, 1975.

[Kleinrock 1975b] L. Kleinrock, F. A. Tobagi, "Packet Switching in Radio Channels: Part I—Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Transactions on Communications*, Vol. 23, No. 12 (Dec. 1975), pp. 1400–1416.

[Kleinrock 1976] L. Kleinrock, *Queuing Systems, Vol. 2*, John Wiley, New York, 1976.

[Kleinrock 2004] L. Kleinrock, "The Birth of the Internet," http://www.lk.cs.ucla.edu/LK/Inet/birth.html

[Kohler 2006] E. Kohler, M. Handley, S. Floyd, "DDCP: Designing DCCP: Congestion Control Without Reliability," *Proc. 2006 ACM SIGCOMM* (Pisa, Italy, Sept. 2006).

[Kolding 2003] T. Kolding, K. Pedersen, J. Wigard, F. Frederiksen, P. Mogensen, "High Speed Downlink Packet Access: WCDMA Evolution," *IEEE Vehicular Technology Society News* (Feb. 2003), pp. 4–10.

[Koponen 2010] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, S. Shenker, "Onix: A Distributed Control Platform for Large-Scale Production Networks," *9th USENIX conference on Operating systems design and implementation (OSDI'10)*, pp. 1–6.

[Koponen 2011] T. Koponen, S. Shenker, H. Balakrishnan, N. Feamster, I. Ganichev, A. Ghodsi, P. B. Godfrey, N. McKeown, G. Parulkar, B. Raghavan, J. Rexford, S. Arianfar, D. Kuptsov, "Architecting for Innovation," *ACM Computer Communications Review*, 2011.

[Korhonen 2003] J. Korhonen, *Introduction to 3G Mobile Communications*, 2nd ed., Artech House, 2003.

[Koziol 2003] J. Koziol, *Intrusion Detection with Snort*, Sams Publishing, 2003.

[Kreutz 2015] D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, Vol. 103, No. 1 (Jan. 2015), pp. 14-76. This paper is also being updated at https://github.com/SDN-Survey/latex/wiki

[Krishnamurthy 2001] B. Krishnamurthy, J. Rexford, *Web Protocols and Practice: HTTP/ 1.1, Networking Protocols, and Traffic Measurement*, Addison-Wesley, Boston, MA, 2001.

[Kulkarni 2005] S. Kulkarni, C. Rosenberg, "Opportunistic Scheduling: Generalizations to Include Multiple Constraints, Multiple Interfaces, and Short Term Fairness," *Wireless Networks*, 11 (2005), 557–569.

[Kumar 2006] R. Kumar, K.W. Ross, "Optimal Peer-Assisted File Distribution: Single and Multi-Class Problems," *IEEE Workshop on Hot Topics in Web Systems and Technologies* (Boston, MA, 2006).

[Labovitz 1997] C. Labovitz, G. R. Malan, F. Jahanian, "Internet Routing Instability," *Proc. 1997 ACM SIGCOMM* (Cannes, France, Sept. 1997), pp. 115–126.

[Labovitz 2010] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, "Internet Inter-Domain Traffic," *Proc. 2010 ACM SIGCOMM*.

[Labrador 1999] M. Labrador, S. Banerjee, "Packet Dropping Policies for ATM and IP Networks," *IEEE Communications Surveys*, Vol. 2, No. 3 (Third Quarter 1999), pp. 2–14.

[Lacage 2004] M. Lacage, M.H. Manshaei, T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," *ACM Int. Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)* (Venice, Italy, Oct. 2004).

[Lakhina 2004] A. Lakhina, M. Crovella, C. Diot, "Diagnosing Network-Wide Traffic Anomalies," *Proc. 2004 ACM SIGCOMM*.

[Lakhina 2005] A. Lakhina, M. Crovella, C. Diot, "Mining Anomalies Using Traffic Feature Distributions," *Proc. 2005 ACM SIGCOMM*.

[Lakshman 1997] T. V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 3 (1997), pp. 336–350.

[Lakshman 2004] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, T. Woo, "The SoftRouter Architecture," *Proc. 3nd ACM Workshop on Hot Topics in Networks (Hotnets-III)*, Nov. 2004.

[Lam 1980] S. Lam, "A Carrier Sense Multiple Access Protocol for Local Net-

works," *Computer Networks*, Vol. 4 (1980), pp. 21–32.

[Lamport 1989] L. Lamport, "The Part-Time Parliament," Technical Report 49, Systems Research Center, Digital Equipment Corp., Palo Alto, Sept. 1989.

[Lampson 1983] Lampson, Butler W. "Hints for computer system design," *ACM SIGOPS Operating Systems Review*, Vol. 17, No. 5, 1983.

[Lampson 1996] B. Lampson, "How to Build a Highly Available System Using Consensus," *Proc. 10th International Workshop on Distributed Algorithms* (WDAG '96), Özalp Babaoglu and Keith Marzullo (Eds.), Springer-Verlag, pp. 1–17.

[Lawton 2001] G. Lawton, "Is IPv6 Finally Gaining Ground?" *IEEE Computer Magazine* (Aug. 2001), pp. 11–15.

[LeBlond 2011] S. Le Blond, C. Zhang, A. Legout, K. Ross, W. Dabbous. 2011, "I know where you are and what you are sharing: exploiting P2P communications to invade users' privacy." *2011 ACM Internet Measurement Conference*, ACM, New York, NY, USA, pp. 45–60.

[Leighton 2009] T. Leighton, "Improving Performance on the Internet," *Communications of the ACM*, Vol. 52, No. 2 (Feb. 2009), pp. 44–51.

[Leiner 1998] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, S. Woolf, "A Brief History of the Internet," http://www.isoc.org/internet/history/brief.html

[Leung 2006] K. Leung, V. O.K. Li, "TCP in Wireless Networks: Issues, Approaches, and Challenges," *IEEE Commun. Surveys and Tutorials*, Vol. 8, No. 4 (2006), pp. 64–79.

[Levin 2012] D. Levin, A. Wundsam, B. Heller, N. Handigol, A. Feldmann, "Logically Centralized?: State Distribution Trade-offs in Software Defined Networks," *Proc. First Workshop on Hot Topics in Software Defined Networks* (Aug. 2012), pp. 1–6.

[Li 2004] L. Li, D. Alderson, W. Willinger, J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-Level Topology," *Proc. 2004 ACM SIGCOMM* (Portland, OR, Aug. 2004).

[Li 2007] J. Li, M. Guidero, Z. Wu, E. Purpus, T. Ehrenkranz, "BGP Routing Dynamics Revisited." *ACM Computer Communication Review* (Apr. 2007).

[Li 2015] S.Q. Li, "Building Softcom Ecosystem Foundation," Open Networking Summit, 2015.

[Lin 2001] Y. Lin, I. Chlamtac, *Wireless and Mobile Network Architectures*, John Wiley and Sons, New York, NY, 2001.

[Liogkas 2006] N. Liogkas, R. Nelson, E. Kohler, L. Zhang, "Exploiting BitTorrent for Fun (but Not Profit)," *6th International Workshop on Peer-to-Peer Systems (IPTPS 2006).*

[Liu 2003] J. Liu, I. Matta, M. Crovella, "End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment," *Proc. WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks.*

[Locher 2006] T. Locher, P. Moor, S. Schmid, R. Wattenhofer, "Free Riding in BitTorrent is Cheap," *Proc. ACM HotNets 2006* (Irvine CA, Nov. 2006).

[Lui 2004] J. Lui, V. Misra, D. Rubenstein, "On the Robustness of Soft State Protocols," *Proc. IEEE Int. Conference on Network Protocols (ICNP '04)*, pp. 50–60.

[Mahdavi 1997] J. Mahdavi, S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," unpublished note (Jan. 1997).

[MaxMind 2016] http://www.maxmind.com/app/ip-location

[Maymounkov 2002] P. Maymounkov, D. Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric." *Proceedings of the 1st International Workshop on Peerto-Peer Systems (IPTPS '02)* (Mar. 2002), pp. 53–65.

[McKeown 1997a] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, M. Horowitz, "The Tiny Tera: A Packet Switch Core," *IEEE Micro Magazine* (Jan.–Feb. 1997).

[McKeown 1997b] N. McKeown, "A Fast Switched Backplane for a Gigabit Switched Router," *Business Communications Review*, Vol. 27, No. 12. http://tiny-tera.stanford.edu/~nickm/papers/cisco_fasts_wp.pdf

[McKeown 2008] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner. 2008. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (Mar. 2008), pp. 69–74.

[McQuillan 1980] J. McQuillan, I. Richer, E. Rosen, "The New Routing Algorithm for the Arpanet," *IEEE Transactions on Communications*, Vol. 28, No. 5 (May 1980), pp. 711–719.

[Metcalfe 1976] R. M. Metcalfe, D. R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the Association for Computing Machinery*, Vol. 19, No. 7 (July 1976), pp. 395–404.

[Meyers 2004] A. Myers, T. Ng, H. Zhang, "Rethinking the Service Model: Scaling Ethernet to a Million Nodes," *ACM Hotnets Conference*, 2004.

[MFA Forum 2016] IP/MPLS Forum homepage, http://www.ipmplsforum.org/

[Mockapetris 1988] P. V. Mockapetris, K. J. Dunlap, "Development of the Domain Name System," *Proc. 1988 ACM SIGCOMM* (Stanford, CA, Aug. 1988).

[Mockapetris 2005] P. Mockapetris, Sigcomm Award Lecture, video available at http://www.postel.org/sigcomm

[Molinero-Fernandez 2002] P. Molinaro-Fernandez, N. McKeown, H. Zhang, "Is IP Going to Take Over the World (of Communications)?" *Proc. 2002 ACM Hotnets.*

[Molle 1987] M. L. Molle, K. Sohraby, A. N. Venetsanopoulos, "Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 5, No. 6 (1987), pp. 956–968.

[Moore 2001] D. Moore, G. Voelker, S. Savage, "Inferring Internet Denial of Service Activity," *Proc. 2001 USENIX Security Symposium* (Washington, DC, Aug. 2001).

[Motorola 2007] Motorola, "Long Term Evolution (LTE): A Technical Overview," http://www.motorola.com/staticfiles/Business/Solutions/Industry%20Solutions/Service%20Providers/Wireless%20Operators/LTE/_Document/Static%20Files/6834_MotDoc_New.pdf

[Mouly 1992] M. Mouly, M. Pautet, *The GSM System for Mobile Communications*, Cell and Sys, Palaiseau, France, 1992.

[Moy 1998] J. Moy, *OSPF: Anatomy of An Internet Routing Protocol*, Addison-Wesley, Reading, MA, 1998.

[Mukherjee 1997] B. Mukherjee, *Optical Communication Networks*, McGraw-Hill, 1997.

[Mukherjee 2006] B. Mukherjee, *Optical WDM Networks*, Springer, 2006.

[Mysore 2009] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, A. Vahdat, "PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," *Proc. 2009 ACM SIGCOMM.*

[Nahum 2002] E. Nahum, T. Barzilai, D. Kandlur, "Performance Issues in WWW Servers," *IEEE/ACM Transactions on Networking*, Vol 10, No. 1 (Feb. 2002).

[Netflix Open Connect 2016] Netflix Open Connect CDN, 2016, https://openconnect.netflix.com/

[Netflix Video 1] Designing Netflix's Content Delivery System, D. Fulllager, 2014, https://www.youtube.com/watch?v=LkLLpYdDINA

[Netflix Video 2] Scaling the Netflix Global CDN, D. Temkin, 2015, https://www.youtube.com/watch?v=tbqcsHg-Q_o

[Neumann 1997] R. Neumann, "Internet Routing Black Hole," *The Risks Digest: Forum on Risks to the Public in Computers and Related Systems*, Vol. 19, No. 12 (May 1997). http://catless.ncl.ac.uk/Risks/19.12.html#subj1.1

[Neville-Neil 2009] G. Neville-Neil, "Whither Sockets?" *Communications of the ACM*, Vol. 52, No. 6 (June 2009), pp. 51–55.

[Nicholson 2006] A Nicholson, Y. Chawathe, M. Chen, B. Noble, D. Wetherall, "Improved Access Point Selection," *Proc. 2006 ACM Mobisys Conference* (Uppsala Sweden, 2006).

[Nielsen 1997] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, C. Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG," *W3C Document*, 1997 (also appears in *Proc. 1997 ACM SIGCOM* (Cannes, France, Sept 1997), pp. 155–166.

[NIST 2001] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Federal Information Processing Standards 197, Nov. 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[NIST IPv6 2015] US National Institute of Standards and Technology, "Estimating IPv6 & DNSSEC Deployment SnapShots," http://fedv6-deployment.antd.nist.gov/snap-all.html

[Nmap 2012] Nmap homepage, http://www.insecure.com/nmap

[Nonnenmacher 1998] J. Nonnenmacher, E. Biersak, D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4 (Aug. 1998), pp. 349–361.

[Nygren 2010] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun, "The Akamai Network: A Platform for High-performance Internet Applications," *SIGOPS Oper. Syst. Rev.* 44, 3 (Aug. 2010), 2–19.

[ONF 2016] Open Networking Foundation, Technical Library, https://www.opennetworking.org/sdn-resources/technical-library

[ONOS 2016] Open Network Operating System (ONOS), "Architecture Guide," https://wiki.onosproject.org/display/ONOS/Architecture+Guide, 2016.

[OpenFlow 2009] Open Network Foundation, "OpenFlow Switch Specification 1.0.0, TS-001," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

[OpenDaylight Lithium 2016] OpenDaylight, "Lithium," https://www.opendaylight.org/lithium

[OSI 2012] International Organization for Standardization homepage, http://www.iso.org/iso/en/ISOOnline.frontpage

[Osterweil 2012] E. Osterweil, D. McPherson, S. DiBenedetto, C. Papadopoulos, D. Massey, "Behavior of DNS Top Talkers," *Passive and Active Measurement Conference*, 2012.

[Padhye 2000] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Transactions on Networking*, Vol. 8 No. 2 (Apr. 2000), pp. 133–145.

[Padhye 2001] J. Padhye, S. Floyd, "On Inferring TCP Behavior," *Proc. 2001 ACM SIGCOMM* (San Diego, CA, Aug. 2001).

[Palat 2009] S. Palat, P. Godin, "The LTE Network Architecture: A Comprehensive Tutorial," in *LTE—The UMTS Long Term Evolution: From Theory to Practice.* Also available as a standalone Alcatel white paper.

[Panda 2013] A. Panda, C. Scott, A. Ghodsi, T. Koponen, S. Shenker, "CAP for Networks," *Proc. ACM HotSDN '13*, pp. 91–96.

[Parekh 1993] A. Parekh, R. Gallagher, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3 (June 1993), pp. 344–357.

[Partridge 1992] C. Partridge, S. Pink, "An Implementation of the Revised Internet Stream Protocol (ST-2)," *Journal of Internetworking: Research and Experience*, Vol. 3, No. 1 (Mar. 1992).

[Partridge 1998] C. Partridge, et al. "A Fifty Gigabit per second IP Router," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 3 (Jun. 1998), pp. 237–248.

[Pathak 2010] A. Pathak, Y. A. Wang, C. Huang, A. Greenberg, Y. C. Hu, J. Li, K. W. Ross, "Measuring and Evaluating TCP Splitting for Cloud Services," *Passive and Active Measurement (PAM) Conference* (Zurich, 2010).

[Perkins 1994] A. Perkins, "Networking with Bob Metcalfe," *The Red Herring Magazine* (Nov. 1994).

[Perkins 1998] C. Perkins, O. Hodson, V. Hardman, "A Survey of Packet Loss Recovery Techniques for Streaming Audio," *IEEE Network Magazine* (Sept./Oct. 1998), pp. 40–47.

[Perkins 1998b] C. Perkins, *Mobile IP: Design Principles and Practice*, Addison-Wesley, Reading, MA, 1998.

[Perkins 2000] C. Perkins, *Ad Hoc Networking*, Addison-Wesley, Reading, MA, 2000.

[Perlman 1999] R. Perlman, *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, 2nd ed., Addison-Wesley Professional Computing Series, Reading, MA, 1999.

[PGPI 2016] The International PGP homepage, http://www.pgpi.org

[Phifer 2000] L. Phifer, "The Trouble with NAT," *The Internet Protocol Journal*, Vol. 3, No. 4 (Dec. 2000), http://www.cisco.com/warp/public/759/ipj_3-4/ipj_3-4_nat.html

[Piatek 2007] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani, "Do Incentives Build Robustness in Bittorrent?," *Proc. NSDI* (2007).

[Piatek 2008] M. Piatek, T. Isdal, A. Krishnamurthy, T. Anderson, "One Hop Reputations for Peer-to-peer File Sharing Workloads," *Proc. NSDI* (2008).

[Pickholtz 1982] R. Pickholtz, D. Schilling, L. Milstein, "Theory of Spread Spectrum Communication—a Tutorial," *IEEE Transactions on Communications*, Vol. 30, No. 5 (May 1982), pp. 855–884.

[PingPlotter 2016] PingPlotter homepage, http://www.pingplotter.com

[Piscatello 1993] D. Piscatello, A. Lyman Chapin, *Open Systems Networking*, Addison-Wesley, Reading, MA, 1993.

[Pomeranz 2010] H. Pomeranz, "Practical, Visual, Three-Dimensional Pedagogy for Internet Protocol Packet Header Control Fields," https://righteousit.wordpress.com/2010/06/27/practical-visual-three-dimensional-pedagogy-for-internet-protocol-packet-header-control-fields/, June 2010.

[Potaroo 2016] "Growth of the BGP Table–1994 to Present," http://bgp.potaroo.net/

[PPLive 2012] PPLive homepage, http://www.pplive.com

[Qazi 2013] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," ACM *SIGCOMM Conference* (Aug. 2013), pp. 27–38.

[Quagga 2012] Quagga, "Quagga Routing Suite," http://www.quagga.net/

[Quittner 1998] J. Quittner, M. Slatalla, *Speeding the Net: The Inside Story of Netscape and How It Challenged Microsoft*, Atlantic Monthly Press, 1998.

[Quova 2016] www.quova.com

[Ramakrishnan 1990] K. K. Ramakrishnan, R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks," *ACM Transactions on Computer Systems*, Vol. 8, No. 2 (May 1990), pp. 158–181.

[Raman 1999] S. Raman, S. McCanne, "A Model, Analysis, and Protocol Framework for Soft State-based Communication," *Proc. 1999 ACM SIGCOMM* (Boston, MA, Aug. 1999).

[Raman 2007] B. Raman, K. Chebrolu, "Experiences in Using WiFi for Rural Internet in India," *IEEE Communications Magazine*, Special Issue on New Directions in Networking Technologies in Emerging Economies (Jan. 2007).

[Ramaswami 2010] R. Ramaswami, K. Sivarajan, G. Sasaki, *Optical Networks: A Practical Perspective*, Morgan Kaufman Publishers, 2010.

[Ramjee 1994] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks," *Proc. 1994 IEEE INFOCOM.*

[Rao 2011] A. S. Rao, Y. S. Lim, C. Barakat, A. Legout, D. Towsley, W. Dabbous, "Network Characteristics of Video Streaming Traffic," *Proc. 2011 ACM CoNEXT* (Tokyo).

[Ren 2006] S. Ren, L. Guo, X. Zhang, "ASAP: An AS-Aware Peer-Relay Protocol for High Quality VoIP," *Proc. 2006 IEEE ICDCS* (Lisboa, Portugal, July 2006).

[Rescorla 2001] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, Boston, 2001.

[RFC 001] S. Crocker, "Host Software," RFC 001 (the *very first* RFC!).

[RFC 768] J. Postel, "User Datagram Protocol," RFC 768, Aug. 1980.

[RFC 791] J. Postel, "Internet Protocol: DARPA Internet Program Protocol Specification," RFC 791, Sept. 1981.

[RFC 792] J. Postel, "Internet Control Message Protocol," RFC 792, Sept. 1981.

[RFC 793] J. Postel, "Transmission Control Protocol," RFC 793, Sept. 1981.

[RFC 801] J. Postel, "NCP/TCP Transition Plan," RFC 801, Nov. 1981.

[RFC 826] D. C. Plummer, "An Ethernet Address Resolution Protocol—or—Converting Network Protocol Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware," RFC 826, Nov. 1982.

[RFC 829] V. Cerf, "Packet Satellite Technology Reference Sources," RFC 829, Nov. 1982.

[RFC 854] J. Postel, J. Reynolds, "TELNET Protocol Specification," RFC 854, May 1993.

[RFC 950] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure," RFC 950, Aug. 1985.

[RFC 959] J. Postel and J. Reynolds, "File Transfer Protocol (FTP)," RFC 959, Oct. 1985.

[RFC 1034] P. V. Mockapetris, "Domain Names—Concepts and Facilities," RFC 1034, Nov. 1987.

[RFC 1035] P. Mockapetris, "Domain Names—Implementation and Specification," RFC 1035, Nov. 1987.

[RFC 1058] C. L. Hendrick, "Routing Information Protocol," RFC 1058, June 1988.

[RFC 1071] R. Braden, D. Borman, and C. Partridge, "Computing the Internet Checksum," RFC 1071, Sept. 1988.

[RFC 1122] R. Braden, "Requirements for Internet Hosts—Communication Layers," RFC 1122, Oct. 1989.

[RFC 1123] R. Braden, ed., "Requirements for Internet Hosts—Application and Support," RFC-1123, Oct. 1989.

[RFC 1142] D. Oran, "OSI IS-IS Intra-Domain Routing Protocol," RFC 1142, Feb. 1990.

[RFC 1190] C. Topolcic, "Experimental Internet Stream Protocol: Version 2 (ST-II)," RFC 1190, Oct. 1990.

[RFC 1256] S. Deering, "ICMP Router Discovery Messages," RFC 1256, Sept. 1991.

[RFC 1320] R. Rivest, "The MD4 Message-Digest Algorithm," RFC 1320, Apr. 1992.

[RFC 1321] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, Apr. 1992.

[RFC 1323] V. Jacobson, S. Braden, D. Borman, "TCP Extensions for High Performance," RFC 1323, May 1992.

[RFC 1422] S. Kent, "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management," RFC 1422.

[RFC 1546] C. Partridge, T. Mendez, W. Milliken, "Host Anycasting Service," RFC 1546, 1993.

[RFC 1584] J. Moy, "Multicast Extensions to OSPF," RFC 1584, Mar. 1994.

[RFC 1633] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.

[RFC 1636] R. Braden, D. Clark, S. Crocker, C. Huitema, "Report of IAB Workshop on Security in the Internet Architecture," RFC 1636, Nov. 1994.

[RFC 1700] J. Reynolds, J. Postel, "Assigned Numbers," RFC 1700, Oct. 1994.

[RFC 1752] S. Bradner, A. Mankin, "The Recommendations for the IP Next Generation Protocol," RFC 1752, Jan. 1995.

[RFC 1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear,

"Address Allocation for Private Internets," RFC 1918, Feb. 1996.

[RFC 1930] J. Hawkinson, T. Bates, "Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)," RFC 1930, Mar. 1996.

[RFC 1939] J. Myers, M. Rose, "Post Office Protocol—Version 3," RFC 1939, May 1996.

[RFC 1945] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol—HTTP/1.0," RFC 1945, May 1996.

[RFC 2003] C. Perkins, "IP Encapsulation Within IP," RFC 2003, Oct. 1996.

[RFC 2004] C. Perkins, "Minimal Encapsulation Within IP," RFC 2004, Oct. 1996.

[RFC 2018] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, Oct. 1996.

[RFC 2131] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131, Mar. 1997.

[RFC 2136] P. Vixie, S. Thomson, Y. Rekhter, J. Bound, "Dynamic Updates in the Domain Name System," RFC 2136, Apr. 1997.

[RFC 2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification," RFC 2205, Sept. 1997.

[RFC 2210] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210, Sept. 1997.

[RFC 2211] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," RFC 2211, Sept. 1997.

[RFC 2215] S. Shenker, J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements," RFC 2215, Sept. 1997.

[RFC 2326] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, Apr. 1998.

[RFC 2328] J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998.

[RFC 2420] H. Kummert, "The PPP Triple-DES Encryption Protocol (3DESE)," RFC 2420, Sept. 1998.

[RFC 2453] G. Malkin, "RIP Version 2," RFC 2453, Nov. 1998.

[RFC 2460] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Dec. 1998.

[RFC 2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.

[RFC 2578] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)," RFC 2578, Apr. 1999.

[RFC 2579] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Textual Conventions for SMIv2," RFC 2579, Apr. 1999.

[RFC 2580] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Conformance Statements for SMIv2," RFC 2580, Apr. 1999.

[RFC 2597] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.

[RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, R. Fielding, "Hypertext Transfer Protocol—HTTP/1.1," RFC 2616,

June 1999.

[RFC 2663] P. Srisuresh, M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663.

[RFC 2702] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, "Requirements for Traffic Engineering Over MPLS," RFC 2702, Sept. 1999.

[RFC 2827] P. Ferguson, D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," RFC 2827, May 2000.

[RFC 2865] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," RFC 2865, June 2000.

[RFC 3007] B. Wellington, "Secure Domain Name System (DNS) Dynamic Update," RFC 3007, Nov. 2000.

[RFC 3022] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, Jan. 2001.

[RFC 3022] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, Jan. 2001.

[RFC 3031] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001.

[RFC 3032] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, A. Conta, "MPLS Label Stack Encoding," RFC 3032, Jan. 2001.

[RFC 3168] K. Ramakrishnan, S. Floyd, D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, Sept. 2001.

[RFC 3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, Dec. 2001.

[RFC 3221] G. Huston, "Commentary on Inter-Domain Routing in the Internet," RFC 3221, Dec. 2001.

[RFC 3232] J. Reynolds, "Assigned Numbers: RFC 1700 Is Replaced by an On-line Database," RFC 3232, Jan. 2002.

[RFC 3234] B. Carpenter, S. Brim, "Middleboxes: Taxonomy and Issues," RFC 3234, Feb. 2002.

[RFC 3246] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)," RFC 3246, Mar. 2002.

[RFC 3260] D. Grossman, "New Terminology and Clarifications for Diffserv," RFC 3260, Apr. 2002.

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Carmarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, July 2002.

[RFC 3272] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W. S. Lai, "Overview and Principles of Internet Traffic Engineering," RFC 3272, May 2002.

[RFC 3286] L. Ong, J. Yoakum, "An Introduction to the Stream Control Transmission Protocol (SCTP)," RFC 3286, May 2002.

[RFC 3346] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W. S. Lai, "Applicability Statement for Traffic Engineering with MPLS," RFC 3346, Aug. 2002.

[RFC 3390] M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window," RFC 3390, Oct. 2002.

[RFC 3410] J. Case, R. Mundy, D. Partain, "Introduction and Applicability Statements for Internet Standard Management Framework," RFC 3410, Dec. 2002.

[RFC 3414] U. Blumenthal and B. Wijnen, "User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 3414, Dec. 2002.

[RFC 3416] R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)," Dec. 2002.

[RFC 3439] R. Bush, D. Meyer, "Some Internet Architectural Guidelines and Philosophy," RFC 3439, Dec. 2003.

[RFC 3447] J. Jonsson, B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1," RFC 3447, Feb. 2003.

[RFC 3468] L. Andersson, G. Swallow, "The Multiprotocol Label Switching (MPLS) Working Group Decision on MPLS Signaling Protocols," RFC 3468, Feb. 2003.

[RFC 3469] V. Sharma, Ed., F. Hellstrand, Ed, "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery," RFC 3469, Feb. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3469.txt

[RFC 3501] M. Crispin, "Internet Message Access Protocol—Version 4rev1," RFC 3501, Mar. 2003.

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[RFC 3588] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, "Diameter Base Protocol," RFC 3588, Sept. 2003.

[RFC 3649] S. Floyd, "HighSpeed TCP for Large Congestion Windows," RFC 3649, Dec. 2003.

[RFC 3746] L. Yang, R. Dantu, T. Anderson, R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework," Internet, RFC 3746, Apr. 2004.

[RFC 3748] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)," RFC 3748, June 2004.

[RFC 3782] S. Floyd, T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 3782, Apr. 2004.

[RFC 4213] E. Nordmark, R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers," RFC 4213, Oct. 2005.

[RFC 4271] Y. Rekhter, T. Li, S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006.

[RFC 4272] S. Murphy, "BGP Security Vulnerabilities Analysis," RFC 4274, Jan. 2006.

[RFC 4291] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture," RFC 4291, Feb. 2006.

[RFC 4340] E. Kohler, M. Handley, S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, Mar. 2006.

[RFC 4443] A. Conta, S. Deering, M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,"

RFC 4443, Mar. 2006.

[RFC 4346] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," RFC 4346, Apr. 2006.

[RFC 4514] K. Zeilenga, Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names," RFC 4514, June 2006.

[RFC 4601] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, "Protocol Independent Multicast—Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601, Aug. 2006.

[RFC 4632] V. Fuller, T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan," RFC 4632, Aug. 2006.

[RFC 4960] R. Stewart, ed., "Stream Control Transmission Protocol," RFC 4960, Sept. 2007.

[RFC 4987] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987, Aug. 2007.

[RFC 5000] RFC editor, "Internet Official Protocol Standards," RFC 5000, May 2008.

[RFC 5109] A. Li (ed.), "RTP Payload Format for Generic Forward Error Correction," RFC 5109, Dec. 2007.

[RFC 5216] D. Simon, B. Aboba, R. Hurst, "The EAP-TLS Authentication Protocol," RFC 5216, Mar. 2008.

[RFC 5218] D. Thaler, B. Aboba, "What Makes for a Successful Protocol?," RFC 5218, July 2008.

[RFC 5321] J. Klensin, "Simple Mail Transfer Protocol," RFC 5321, Oct. 2008.

[RFC 5322] P. Resnick, Ed., "Internet Message Format," RFC 5322, Oct. 2008.

[RFC 5348] S. Floyd, M. Handley, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 5348, Sept. 2008.

[RFC 5389] J. Rosenberg, R. Mahy, P. Matthews, D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389, Oct. 2008.

[RFC 5411] J Rosenberg, "A Hitchhiker's Guide to the Session Initiation Protocol (SIP)," RFC 5411, Feb. 2009.

[RFC 5681] M. Allman, V. Paxson, E. Blanton, "TCP Congestion Control," RFC 5681, Sept. 2009.

[RFC 5944] C. Perkins, Ed., "IP Mobility Support for IPv4, Revised," RFC 5944, Nov. 2010.

[RFC 6265] A Barth, "HTTP State Management Mechanism," RFC 6265, Apr. 2011.

[RFC 6298] V. Paxson, M. Allman, J. Chu, M. Sargent, "Computing TCP's Retransmission Timer," RFC 6298, June 2011.

[RFC 7020] R. Housley, J. Curran, G. Huston, D. Conrad, "The Internet Numbers Registry System," RFC 7020, Aug. 2013.

[RFC 7094] D. McPherson, D. Oran, D. Thaler, E. Osterweil, "Architectural Considerations of IP Anycast," RFC 7094, Jan. 2014.

[RFC 7323] D. Borman, R. Braden, V. Jacobson, R. Scheffenegger (ed.), "TCP Extensions for High Performance," RFC 7323, Sept. 2014.

[RFC 7540] M. Belshe, R. Peon, M. Thomson (Eds), "Hypertext Transfer Protocol

Version 2 (HTTP/2)," RFC 7540, May 2015.

[Richter 2015] P. Richter, M. Allman, R. Bush, V. Paxson, "A Primer on IPv4 Scarcity," *ACM SIGCOMM Computer Communication Review*, Vol. 45, No. 2 (Apr. 2015), pp. 21–32.

[Roberts 1967] L. Roberts, T. Merril, "Toward a Cooperative Network of Time-Shared Computers," *AFIPS Fall Conference* (Oct. 1966).

[Rodriguez 2010] R. Rodrigues, P. Druschel, "Peer-to-Peer Systems," *Communications of the ACM*, Vol. 53, No. 10 (Oct. 2010), pp. 72–82.

[Rohde 2008] Rohde, Schwarz, "UMTS Long Term Evolution (LTE) Technology Introduction," Application Note 1MA111.

[Rom 1990] R. Rom, M. Sidi, *Multiple Access Protocols: Performance and Analysis*, Springer-Verlag, New York, 1990.

[Root Servers 2016] Root Servers home page, http://www.root-servers.org/

[RSA 1978] R. Rivest, A. Shamir, L. Adelman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM*, Vol. 21, No. 2 (Feb. 1978), pp. 120–126.

[RSA Fast 2012] RSA Laboratories, "How Fast Is RSA?" http://www.rsa.com/rsalabs/node.asp?id=2215

[RSA Key 2012] RSA Laboratories, "How Large a Key Should Be Used in the RSA Crypto System?" http://www.rsa.com/rsalabs/node.asp?id=2218

[Rubenstein 1998] D. Rubenstein, J. Kurose, D. Towsley, "Real-Time Reliable Multicast Using Proactive Forward Error Correction," *Proceedings of NOSSDAV '98* (Cambridge, UK, July 1998).

[Ruiz-Sanchez 2001] M. Ruiz-Sánchez, E. Biersack, W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," *IEEE Network Magazine*, Vol. 15, No. 2 (Mar./Apr. 2001), pp. 8–23.

[Saltzer 1984] J. Saltzer, D. Reed, D. Clark, "End-to-End Arguments in System Design," *ACM Transactions on Computer Systems (TOCS)*, Vol. 2, No. 4 (Nov. 1984).

[Sandvine 2015] "Global Internet Phenomena Report, Spring 2011," http://www.sandvine.com/news/global broadband trends.asp, 2011.

[Sardar 2006] B. Sardar, D. Saha, "A Survey of TCP Enhancements for Last-Hop Wireless Networks," *IEEE Commun. Surveys and Tutorials*, Vol. 8, No. 3 (2006), pp. 20–34.

[Saroiu 2002] S. Saroiu, P. K. Gummadi, S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. of Multimedia Computing and Networking (MMCN)* (2002).

[Sauter 2014] M. Sauter, *From GSM to LTE-Advanced*, John Wiley and Sons, 2014.

[Savage 2015] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, R. White, "Enhanced Interior Gateway Routing Protocol," Internet Draft, draft-savage-eigrp-04.txt, Aug. 2015.

[Saydam 1996] T. Saydam, T. Magedanz, "From Networks and Network Management into Service and Service Management," *Journal of Networks and System Management*, Vol. 4, No. 4 (Dec. 1996), pp. 345–348.

[Schiller 2003] J. Schiller, *Mobile Communications* 2nd edition, Addison Wesley, 2003.

[Schneier 1995] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, 1995.

[Schulzrinne-RTP 2012] Henning Schulzrinne's RTP site, http://www.cs.columbia .edu/~hgs/rtp

[Schulzrinne-SIP 2016] Henning Schulzrinne's SIP site, http://www.cs.columbia. edu/~hgs/sip

[Schwartz 1977] M. Schwartz, *Computer-Communication Network Design and Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[Schwartz 1980] M. Schwartz, *Information, Transmission, Modulation, and Noise*, McGraw Hill, New York, NY 1980.

[Schwartz 1982] M. Schwartz, "Performance Analysis of the SNA Virtual Route Pacing Control," *IEEE Transactions on Communications*, Vol. 30, No. 1 (Jan. 1982), pp. 172–184.

[Scourias 2012] J. Scourias, "Overview of the Global System for Mobile Communications: GSM." http://www.privateline.com/PCS/GSM0.html

[SDNHub 2016] SDNHub, "App Development Tutorials," http://sdnhub.org/ tutorials/

[Segaller 1998] S. Segaller, *Nerds 2.0.1, A Brief History of the Internet*, TV Books, New York, 1998.

[Sekar 2011] V. Sekar, S. Ratnasamy, M. Reiter, N. Egi, G. Shi, " The Middlebox Manifesto: Enabling Innovation in Middlebox Deployment," *Proc. 10th ACM Workshop on Hot Topics in Networks (HotNets)*, Article 21, 6 pages.

[Serpanos 2011] D. Serpanos, T. Wolf, *Architecture of Network Systems*, Morgan Kaufmann Publishers, 2011.

[Shacham 1990] N. Shacham, P. McKenney, "Packet Recovery in High-Speed Networks Using Coding and Buffer Management," *Proc. 1990 IEEE INFOCOM* (San Francisco, CA, Apr. 1990), pp. 124–131.

[Shaikh 2001] A. Shaikh, R. Tewari, M. Agrawal, "On the Effectiveness of DNS-based Server Selection," *Proc. 2001 IEEE INFOCOM*.

[Singh 1999] S. Singh, *The Code Book: The Evolution of Secrecy from Mary, Queen of Scotsto Quantum Cryptography*, Doubleday Press, 1999.

[Singh 2015] A. Singh, J. Ong,. Agarwal, G. Anderson, A. Armistead, R. Banno, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, A. Vahdat, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," Sigcomm, 2015.

[SIP Software 2016] H. Schulzrinne Software Package site, http://www. cs.columbia.edu/IRT/software

[Skoudis 2004] E. Skoudis, L. Zeltser, *Malware: Fighting Malicious Code*, Prentice Hall, 2004.

[Skoudis 2006] E. Skoudis, T. Liston, *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd Edition)*, Prentice Hall, 2006.

[Smith 2009] J. Smith, "Fighting Physics: A Tough Battle," *Communications of the ACM*, Vol. 52, No. 7 (July 2009), pp. 60–65.

[Snort 2012] Sourcefire Inc., Snort homepage, http://http://www.snort.org/

[Solensky 1996] F. Solensky, "IPv4 Address Lifetime Expectations," in *IPng: Internet Protocol Next Generation* (S. Bradner, A. Mankin, ed.), Addison-Wesley, Reading, MA, 1996.

[Spragins 1991] J. D. Spragins, *Telecommunications Protocols and Design*, Addison-Wesley, Reading, MA, 1991.

[Srikant 2004] R. Srikant, *The Mathematics of Internet Congestion Control*, Birkhauser, 2004

[Steinder 2002] M. Steinder, A. Sethi, "Increasing Robustness of Fault Localization Through Analysis of Lost, Spurious, and Positive Symptoms," *Proc. 2002 IEEE INFOCOM.*

[Stevens 1990] W. R. Stevens, *Unix Network Programming*, Prentice-Hall, Englewood Cliffs, NJ.

[Stevens 1994] W. R. Stevens, *TCP/IP Illustrated, Vol. 1: The Protocols*, Addison-Wesley, Reading, MA, 1994.

[Stevens 1997] W.R. Stevens, *Unix Network Programming, Volume 1: Networking APIs-Sockets and XTI*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[Stewart 1999] J. Stewart, *BGP4: Interdomain Routing in the Internet*, Addison-Wesley, 1999.

[Stone 1998] J. Stone, M. Greenwald, C. Partridge, J. Hughes, "Performance of Checksums and CRC's Over Real Data," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 5 (Oct. 1998), pp. 529–543.

[Stone 2000] J. Stone, C. Partridge, "When Reality and the Checksum Disagree," *Proc. 2000 ACM SIGCOMM* (Stockholm, Sweden, Aug. 2000).

[Strayer 1992] W. T. Strayer, B. Dempsey, A. Weaver, *XTP: The Xpress Transfer Protocol*, Addison-Wesley, Reading, MA, 1992.

[Stubblefield 2002] A. Stubblefield, J. Ioannidis, A. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP," *Proceedings of 2002 Network and Distributed Systems Security Symposium* (2002), pp. 17–22.

[Subramanian 2000] M. Subramanian, *Network Management: Principles and Practice*, Addison-Wesley, Reading, MA, 2000.

[Subramanian 2002] L. Subramanian, S. Agarwal, J. Rexford, R. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points," *Proc. 2002 IEEE INFOCOM*.

[Sundaresan 2006] K.Sundaresan, K. Papagiannaki, "The Need for Cross-layer Information in Access Point Selection," *Proc. 2006 ACM Internet Measurement Conference* (Rio De Janeiro, Oct. 2006).

[Suh 2006] K. Suh, D. R. Figueiredo, J. Kurose and D. Towsley, "Characterizing and Detecting Relayed Traffic: A Case Study Using Skype," *Proc. 2006 IEEE INFOCOM* (Barcelona, Spain, Apr. 2006).

[Sunshine 1978] C. Sunshine, Y. Dalal, "Connection Management in Transport Protocols," *Computer Networks*, North-Holland, Amsterdam, 1978.

[Tariq 2008] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, M. Ammar, "Answering What-If Deployment and Configuration Questions with WISE," *Proc. 2008 ACM SIGCOMM* (Aug. 2008).

[TechnOnLine 2012] TechOnLine, "Protected Wireless Networks," online webcast tutorial, http://www.techonline.com/community/tech_topic/internet/21752

[Teixeira 2006] R. Teixeira, J. Rexford, "Managing Routing Disruptions in Internet Service Provider Networks," *IEEE Communications Magazine* (Mar. 2006).

[Think 2012] Technical History of Network Protocols, "Cyclades," http://www.cs.utexas.edu/users/chris/think/Cyclades/index.shtml

[Tian 2012] Y. Tian, R. Dey, Y. Liu, K. W. Ross, "China's Internet: Topology Mapping and Geolocating," *IEEE INFOCOM Mini-Conference 2012* (Orlando, FL, 2012).

[TLD list 2016] TLD list maintained by Wikipedia, https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains

[Tobagi 1990] F. Tobagi, "Fast Packet Switch Architectures for Broadband Integrated Networks," *Proc. 1990 IEEE INFOCOM*, Vol. 78, No. 1 (Jan. 1990), pp. 133–167.

[TOR 2016] Tor: Anonymity Online, http://www.torproject.org

[Torres 2011] R. Torres, A. Finamore, J. R. Kim, M. M. Munafo, S. Rao, "Dissecting Video Server Selection Strategies in the YouTube CDN," *Proc. 2011 Int. Conf. on Distributed Computing Systems.*

[Tourrilhes 2014] J. Tourrilhes, P. Sharma, S. Banerjee, J. Petit, "SDN and Openflow Evolution: A Standards Perspective," *IEEE Computer Magazine*, Nov. 2014, pp. 22–29.

[Turner 1988] J. S. Turner, "Design of a Broadcast packet switching network," *IEEE Transactions on Communications*, Vol. 36, No. 6 (June 1988), pp. 734–743.

[Turner 2012] B. Turner, "2G, 3G, 4G Wireless Tutorial," http://blogs.nmscommunications.com/communications/2008/10/2g-3g-4g-wireless-tutorial.html

[UPnP Forum 2016] UPnP Forum homepage, http://www.upnp.org/

[van der Berg 2008] R. van der Berg, "How the 'Net Works: An Introduction to Peering and Transit," http://arstechnica.com/guides/other/peering-and-transit.ars

[van der Merwe 1998] J. van der Merwe, S. Rooney, I. Leslie, S. Crosby, "The Tempest: A Practical Framework for Network Programmability," *IEEE Network*, Vol. 12, No. 3 (May 1998), pp. 20–28.

[Varghese 1997] G. Varghese, A. Lauck, "Hashed and Hierarchical Timing Wheels: Efficient Data Structures for Implementing a Timer Facility," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6 (Dec. 1997), pp. 824–834.

[Vasudevan 2012] S. Vasudevan, C. Diot, J. Kurose, D. Towsley, "Facilitating Access Point Selection in IEEE 802.11 Wireless Networks," *Proc. 2005 ACM Internet Measurement Conference*, (San Francisco CA, Oct. 2005).

[Villamizar 1994] C. Villamizar, C. Song. "High Performance TCP in ANSNET," *ACM SIGCOMM Computer Communications Review*, Vol. 24, No. 5 (1994), pp. 45–60.

[Viterbi 1995] A. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, Addison-Wesley, Reading, MA, 1995.

[Vixie 2009] P. Vixie, "What DNS Is Not," *Communications of the ACM*, Vol. 52, No. 12 (Dec. 2009), pp. 43–47.

[Wakeman 1992] I. Wakeman, J. Crowcroft, Z. Wang, D. Sirovica, "Layering Considered Harmful," *IEEE Network* (Jan. 1992), pp. 20–24.

[Waldrop 2007] M. Waldrop, "Data Center in a Box," *Scientific American* (July 2007).

[Wang 2004] B. Wang, J. Kurose, P. Shenoy, D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study," *Proc. 2004 ACM Multimedia Conference* (New York, NY, Oct. 2004).

[Wang 2008] B. Wang, J. Kurose, P. Shenoy, D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study," *ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP)*, Vol. 4, No. 2 (Apr. 2008), p. 16. 1–22.

[Wang 2010] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, M. Ryan, "c-Through: Part-time Optics in Data Centers," *Proc. 2010 ACM SIGCOMM*.

[Wei 2006] W. Wei, C. Zhang, H. Zang, J. Kurose, D. Towsley, "Inference and Evaluation of Split-Connection Approaches in Cellular Data Networks," *Proc. Active and Passive Measurement Workshop* (Adelaide, Australia, Mar. 2006).

[Wei 2007] D. X. Wei, C. Jin, S. H. Low, S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Transactions on Networking* (2007).

[Weiser 1991] M. Weiser, "The Computer for the Twenty-First Century," *Scientific American* (Sept. 1991): 94–10. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[White 2011] A. White, K. Snow, A. Matthews, F. Monrose, "Hookt on fon-iks: Phonotactic Reconstruction of Encrypted VoIP Conversations," *IEEE Symposium on Security and Privacy*, Oakland, CA, 2011.

[Wigle.net 2016] Wireless Geographic Logging Engine, http://www.wigle.net

[Wiki Satellite 2016] Satellite Internet access, https://en.wikipedia.org/wiki/Satellite_Internet_access

[Wireshark 2016] Wireshark homepage, http://www.wireshark.org

[Wischik 2005] D. Wischik, N. McKeown, "Part I: Buffer Sizes for Core Routers," *ACM SIGCOMM Computer Communications Review*, Vol. 35, No. 3 (July 2005).

[Woo 1994] T. Woo, R. Bindignavle, S. Su, S. Lam, "SNP: an interface for secure network programming," *Proc. 1994 Summer USENIX* (Boston, MA, June 1994), pp. 45–58.

[Wright 2015] J. Wright, J. *Wireless Security Secrets & Solutions*, 3e, "Hacking Exposed Wireless," McGraw-Hill Education, 2015.

[Wu 2005] J. Wu, Z. M. Mao, J. Rexford, J. Wang, "Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network," *Proc. USENIX NSDI* (2005).

[Xanadu 2012] Xanadu Project homepage, http://www.xanadu.com/

[Xiao 2000] X. Xiao, A. Hannan, B. Bailey, L. Ni, "Traffic Engineering with MPLS in the Internet," *IEEE Network* (Mar./Apr. 2000).

[Xu 2004] L. Xu, K Harfoush, I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks," *IEEE INFOCOM 2004*, pp. 2514–2524.

[Yavatkar 1994] R. Yavatkar, N. Bhagwat, "Improving End-to-End Performance of TCP over Mobile Internetworks," *Proc. Mobile 94 Workshop on Mobile Computing Systems and Applications* (Dec. 1994).

[YouTube 2009] YouTube 2009, Google container data center tour, 2009.

[YouTube 2016] YouTube Statistics, 2016, https://www.youtube.com/yt/press/statistics.html

[Yu 2004] Yu, Fang, H. Katz, Tirunellai V. Lakshman. "Gigabit Rate Packet Pattern-Matching Using TCAM," *Proc. 2004 Int. Conf. Network Protocols*, pp. 174–183.

[Yu 2011] M. Yu, J. Rexford, X. Sun, S. Rao, N. Feamster, "A Survey of VLAN Usage in Campus Networks," *IEEE Communications Magazine*, July 2011.

[Zegura 1997] E. Zegura, K. Calvert, M. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, (Dec. 1997). See also http://www.cc.gatech.edu/projects/gtim for a software package that generates networks with a transit-stub structure.

[Zhang 1993] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource Reservation Protocol," *IEEE Network Magazine*, Vol. 7, No. 9 (Sept. 1993), pp. 8–18.

[Zhang 2007] L. Zhang, "A Retrospective View of NAT," *The IETF Journal*, Vol. 3, Issue 2 (Oct. 2007).

[Zhang 2015] G. Zhang, W. Liu, X. Hei, W. Cheng, "Unreeling Xunlei Kankan: Understanding Hybrid CDN-P2P Video-on-Demand Streaming," *IEEE Transactions on Multimedia*, Vol. 17, No. 2, Feb. 2015.

[Zhang X 2102] X. Zhang, Y. Xu, Y. Liu, Z. Guo, Y. Wang, "Profiling Skype Video Calls: Rate Control and Video Quality," *IEEE INFOCOM* (Mar. 2012).

[Zink 2009] M. Zink, K. Suh, Y. Gu, J. Kurose, "Characteristics of YouTube Network Traffic at a Campus Network—Measurements, Models, and Implications," *Computer Networks*, Vol. 53, No. 4, pp. 501–514, 2009.

计算机网络 自顶向下方法 原书第7版

Computer Networking A Top-Down Approach Seventh Edition

本书自首次出版以来，已被译为14种语言，世界上数百所大学采用本书作为教材，有几十万学生和从业人员利用本书系统学习计算机网络的知识。本书已成为学习计算机网络知识的必读教材之一。

主要特点

- 自顶向下方法。本书采用作者独创的“自顶向下方法”讲授计算机网络的原理及其协议，即从应用层协议开始沿协议栈向下讲解，让读者从实现、应用的角度理解各层的意义，强调应用层范例和应用编程接口，使读者尽快进入每天使用的应用程序环境之中进行学习和创造。
- 以因特网为研究目标。因特网已经变得无所不在，任何网络教科书必须关注因特网。本书以因特网的体系结构和协议为载体讲授基本的计算机网络概念，这有助于激发学生学习网络基本原理的积极性，并理解原理的实际应用。
- 聚焦网络原理。网络领域已经发展得相当成熟，沉淀出很多基础性的重要问题。本书在梳理这些问题的同时，详细介绍了解决问题的方法。用因特网将学生引入网络之门后，再结合这些基础性问题及其解决方案，学生便可以迅速理解几乎任何网络技术。
- 及时更新教学内容。本书根据近年来计算机网络的进展，对内容进行了全面更新，包括用两章的篇幅讨论网络层以反映软件定义网络（SDN）的重要性，删除了FTP和分布式散列表的内容，将ATM网络的内容更新为流行的因特网显式拥塞通告（ECN）内容等。

作者简介

詹姆斯·F. 库罗斯（James F. Kurose） 美国马萨诸塞大学阿默斯特分校计算机科学系教授，研究兴趣包括网络协议和体系结构、网络测量、多媒体通信以及建模和性能评价。由于在教育领域的杰出贡献，他获得了包括IEEE Taylor Booth教育奖章在内的多个教育奖励和荣誉。他是IEEE和ACM会士，还曾担任《IEEE通信会刊》和《IEEE/ACM网络会刊》总编辑。

基思·W. 罗斯（Keith W. Ross） 美国纽约大学（NYU）上海分校工程和计算机科学学院院长以及NYU计算机科学和工程系的Leonard J. Shustek首席教授，研究兴趣包括隐私、社交网络、对等网络、因特网测量、内容分发网络和随机建模。他是IEEE和ACM会士，还曾获得Infocom 2009年优秀论文奖以及《多媒体通信》2011年和2008年优秀论文奖。

Pearson
www.pearson.com

华章教育服务微信号

投稿热线：(010) 88379604
客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

封面设计·锡 彬

上架指导：计算机\网络

ISBN 978-7-111-59971-5

定价：89.00元